Maintenance Scheduling for Modular Systems–Models and Algorithms

by

Eric Jack Zarybnisky

B.S., U.S. Air Force Academy (2001)
S.M., Massachusetts Institute of Technology (2003)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Author .....
Sloan School of Management
August 12, 2011

Certified by .....
Retsef Levi
J. Spencer Standish (1945) Professor of Management
Thesis Supervisor

Certified by .....
Thomas L. Magnanti
Institute Professor
Thesis Supervisor

Accepted by .....
Dimitris Bertsimas
Co-Director, Operations Research Center

Maintenance Scheduling for Modular Systems–Models and Algorithms

by

Eric Jack Zarybnisky

Submitted to the Sloan School of Management
on August 12, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Maintenance scheduling is an integral part of many complex systems. For instance, without effective maintenance scheduling, the combined effects of preventative and corrective maintenance can have severe impacts on the availability of those systems. Based on current Air Force trends including maintenance manpower, dispersed aircraft basing, and increased complexity, there has been a renewed focus on preventative maintenance. To address these concerns, this thesis develops two models for preventative maintenance scheduling for complex systems, the first of interest in the system concept development and design phase, and the second of interest during operations. Both models are highly complex and intractable to solve in their original forms. For the first model, we develop approximation algorithms that yield high quality and easily implementable solutions. To address the second model, we propose a decomposition strategy that produces submodels that can be solved via existing algorithms or via specialized algorithms we develop.

While much of the literature has examined stochastically failing systems, preventative maintenance of *usage limited* systems has received less attention. Of particular interest is the design of modular systems whose components must be repaired/replaced to prevent a failure. By making cost tradeoffs early in development, program managers, designers, engineers, and test conductors can better balance the up front costs associated with system design and testing with the long term cost of maintenance. To facilitate such a tradeoff, the *Modular Maintenance Scheduling Problem* provides a framework for design teams to evaluate different design and operations concepts and then evaluate the long term costs. While the general Modular Maintenance Scheduling Problem does not require maintenance schedules with specific structure, operational considerations push us to consider *cyclic* schedules in which components are maintained at a fixed frequency. In order to efficiently find cyclic schedules, we propose the *Cycle Rounding* algorithm, which has an approximation guarantee of 2, and a family of *Shifted Power-of-Two* algorithms, which have an approximation guarantee of $1/\ln(2) \approx 1.4427$. Computational results indicate that both algorithms perform much better than their associated performance guarantees providing solutions within

15%-25% of a lower bound.

Once a modular system has moved into operations, manpower and transportation scheduling become important considerations when developing maintenance schedules. To address the operations phase, we develop the *Modular Maintenance and System Assembly Model* to balance the tradeoffs between inventory, maintenance capacity, and transportation resources. This model explicitly captures the risk-pooling effects of a central repair facility while also modeling the interaction between repair actions at such a facility. The full model is intractable for all but the smallest instances. Accordingly, we decompose the problem into two parts, the *system assembly* portion and *module repair* portion. Finally, we tie together the Modular Maintenance and System Assembly Model with key concepts from the Modular Maintenance Scheduling Problem to propose an integrated methodology for design and operation.

DISCLAIMER CLAUSE: The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Thesis Supervisor: Retsef Levi
Title: J. Spencer Standish (1945) Professor of Management

Thesis Supervisor: Thomas L. Magnanti
Title: Institute Professor

Acknowledgments

First, I would like to thank my advisors Professor Thomas Magnanti and Retsef Levi. From day one, they immersed themselves in the challenges of Air Force logistics. They pushed me to understand the academic literature as well as the Air Force application. Through generous support, they allowed me to meet with numerous military organizations involved in logistics planning. These trips allowed me to better understand DoD logistics and connect with the professionals involved in the daily work of supporting the military. They also helped me see the applications and implications of our work beyond the military context.

In kind, I would like to acknowledge Professor Stephen Graves as the third member of my dissertation committee. His focus on the applicability and practicality helped focus my research on what really mattered.

I would also like to thank Professor Jack Muckstadt. As a technical advisor and mentor, he always helped me see the brighter side of things. At the same time, he was always willing to question my assumptions and make me justify what I thought. Without his advice, my time here would not have been as much of a growing experience. Even though most of our discussion occurred over the phone or over E-mail, I leave MIT a better officer and academic. His infectious enthusiasm for problem solving has benefited me greatly. I must also thank Jack's wife Linda for opening their home to me where Jack and I worked on models, heuristics, and our ramblings about logistics.

With the focus on Air Force problems, Gen.(ret.) George Babbitt was an invaluable resource with his years of experience. He put me in contact with numerous senior leaders and provided excellent feedback on the trajectory of our research.

To my many friends in the ORC, it's truly been a pleasure. While my time was short, I've made lifelong friendships. Gareth and Nikos, our morning coffee runs were the best way to start my day. To Vishal, thank you for always taking the time to chat, even if you did need to write a few more LPs to get in your daily quota. To my cohort, Matthieu, Ta, Anna, Joline, Chaithanya, Michael, and others, it was great

getting to know you through the trials and tribulations of "Fundamentals", quals, and generals. Nick, it was always a pleasure philosophizing with you about the military and how two young officers thought it should work.

To all of my friends in Natick, thank you for the support you have shown over the past three years. Raising a family while working on a Ph.D. is not an easy task. When I was working late or traveling, you always made sure Emily and the kids were taken care of.

Finally, I could have never made it here without the love and support of my family. When I got home each night, I was always met with a smiling face or a wagging tail. You helped me spend time away from my research which in the end, helped me focus on what really mattered. I look forward to our adventures in D.C. and beyond. Who knows where we'll be living in 5 or 10 years but I know with Emily by my side, it will be a wonderful experience.

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

As technology has developed, the prevalence of modularly designed systems has continued to expand. By allowing modules to be repaired and even replaced with newer technology with minimal impact to other modules, a modular design allows for rapid integration of new technology as well as efficient maintenance planning. In addition, modularly designed systems facilitate preventative maintenance by reducing the risk to other modules in the system when one module is removed for inspection and/or preventative maintenance. This contrasts older systems that required a significant portion of the system to be disassembled in order to perform maintenance.

While modularly designed systems have made maintenance execution easier, maintenance planning and scheduling tools have not kept pace with these changes. In this work we develop models, algorithms, and decision support tools that exploit the modular design of a system to help schedule preventative maintenance. Of particular interest is the specialization that can occur with a modular system. Individual technicians can focus on their particular module or component without a broader understanding of the complete system. While this specialization helps with maintenance efficiency for individual modules and component, it exacerbates manpower issues when they arise. The loss of a maintenance facility can have a similar impact. By considering the long term impacts of near term decisions, our models allow policy makers and maintenance schedules to explore different options to mitigate risk while faced with budgetary and personnel restrictions.

This work primarily focuses on the tradeoffs between design, maintenance resources, inventory, and transportation resources. In all cases, we acknowledge that the operations phase is of primary interest for any complex system. That is, the purpose for a system is to fulfill an operational need. Meeting that operational need should be the focus of system design and any process put in place to support the system. Accordingly, our models and algorithm explicitly capture the operational requirements imposed upon maintenance. While some of these requirements are based on available resources, others are based on the desire to have simple maintenance rules that can be implemented without continuously running complex models and algorithms. By understanding these restrictions, we can better tie design decisions to the operational environment.

1.1 Motivating Examples

A perfect example of the transformation of complex systems is the move from internal combustion engines to jet engines in US Air Force fighter aircraft. This breed of engines, such as the early F100 engines and current F119 and F135 engines, are modularly designed and can be disassembled with relatively little impact between the modules. For instance, the F100 is comprised of five major modules: the inlet/fan module, the core module, the fan drive (low-pressure) turbine module, the augmentor/exhaust module, and the gearbox module, Figure 1-1.

The importance of the F100 family of engines, comprised of the -PW-100, -PW-200, -PW-220/220e, and -PW-229 variants, to the United States Air Force cannot be overstated. With nearly 3,300 engines currently in service, valued at $11.6 billion, supporting the entire F-15 fleet and most of the F-16 fleet, the F100 is flown in more jets than any other Air Force engine [12]. Each year the Department of Defense, primarily the Air Force and Defense Logistics Agency, spends almost $1 billion dollars in spare parts and repair services to support and maintain this fleet of engines [12].

Unlike some other systems on Air Force fighter aircraft, engines are rarely flown until a failure occurs. Rather, due to the potential consequences of a failure, en-

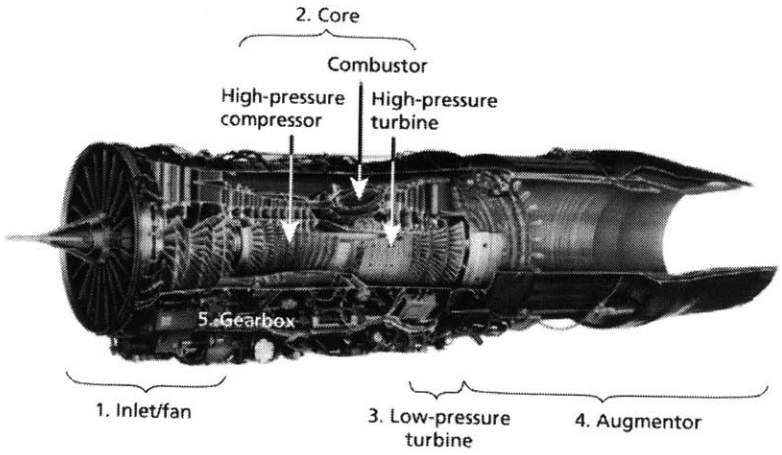

Figure 1-1: Modules of the F100-PW-220 engine [12].

gines undergo extensive preventative maintenance on both calendar and usage driven schedules. These schedules mandate specific types of inspections, maintenance, and replacement. Moreover, these preventative maintenance activities are expensive and require scarce resources.

The fact that preventative maintenance can be, and often is, performed earlier than actual failure occurs, gives rise to an inherent scheduling problem: when to perform specific maintenance actions to meet the preventative maintenance requirements while also minimizing the corresponding costs. Performing several maintenance actions at the same time can reduce manpower requirements, aircraft downtime, and other cost contributions. In conjunction with the scheduling aspects that potentially combine multiple maintenance actions, there may be a number of different maintenance actions for the same component. For instance, metallic components may be inspected visual, via x-ray, or with dye-penetration. Each of these inspections can meet the desired preventative maintenance goals but they do not have the same cost implications nor the same frequency requirements. In many cases, the lower cost actions will need to be performed more frequently to attain the same level of confidence in the equipment. A visual inspection is very quick but can only detect relatively large surface defects. Thus, this inspection must be accomplished more frequently than a dye-penetration examination which can detect much smaller defects.

Another motivating factor for this work is the fact that much of the past research focusing on supporting engines and other modular systems, have focused on setting inventory levels. In the past, correct placement of inventory to support ongoing operations was the primary concern. Limitations surrounding maintenance manpower and transportation were either ignored or assumed sufficient to support the inventory policies that were developed. In recent years with the drawdown in defense manning and the involvement of the United States in conflicts in austere locations, the assumptions of sufficient manpower and a robust transportation network are much less valid. For instance, the total Active Duty strength for the U.S. Air Force has been reduced from almost 900,000 in the late 1960s to approximately 330,000 in 2011 [2]. In addition, transporting supplies and manpower into current combat zones such as Afghanistan is significantly harder than to Europe or the Pacific as was the focus in the mid to late 20th century.

While the motivation for this work is grounded in current Air Force issues, there are numerous other examples of complex and modular systems that require preventative maintenance. One such example is the generators used for electrical power production in hydroelectric dams or other power generation facilities. For such systems, the downtime late at night serves as an opportunity to perform preventative maintenance to ensure a high service levels during times of peak power demand. Another example is large factory production equipment that is preventatively maintained in preparation for high demand seasons.

1.2 Thesis Overview and Contributions

The contributions of this work can be divided into two main areas. Focusing first on the design considerations for modular maintenance, in Chapter 2 we formalize a model that can be used to tradeoff near term design and development costs with long term maintenance costs. During system design and development, decisions about component usable life, maintenance procedures, and indenture depth can have profound impacts upon life cycle maintenance costs. For instance, if a component is designed

with a low usable life, it is difficult to access due to maintenance procedures and/or is deeply indentured (i.e., there are many modules and submodules that must be removed to access the component) this single component can drive high maintenance costs over the life span of the system. These long term costs could be alleviated with short term investment in a longer usable life from redesign of the component, further analysis, or physical testing. By improving maintenance procedures that reduce the time/cost associated with accessing the component, even with a short usable life, life cycle maintenance costs could be reduced. While the impacts of these investments are easily determined for a single component, complex systems with many components and indenture levels are much harder to examine. The *Modular Maintenance Scheduling Problem* models a highly complex system and allows design teams to readily make near term budgetary tradeoffs between components by evaluating the long term impacts of the changes. This model consider critical *cycle limited components* that must be maintained at least as often as a specified frequency. Components are linked by a complex cost structure that allow cost sharing between components. In particular, we consider the general class of non-decreasing submodular cost functions.

While the focus of our first model is on the design phase, we restrict attention to cyclic policies in which maintenance of critical components is performed at fixed frequencies. This restriction ensures the tradeoffs made during design more accurately reflect operational maintenance practices as a cyclic maintenance schedule is much more likely to be codified in practice. In practice, non-cyclic schedules are hard to implement from a viewpoint of resource allocation and planning. In order to produce high quality, cyclic schedules, we develop two approximation algorithms, *cycle rounding* and *shifted power-of-two*, that have not only constant factor worst case guarantees of 2 and $1/\ln(2) \approx 1.4427$ respectively, but also both empirically perform close to optimal. The approximation and empirical performance of these algorithms is in relation to the optimal schedule which may not be cyclic. These results show that the restriction to cyclic policies does not significantly impact the cost of the maintenance schedules. Both algorithms are robust to changes in cycle limits and costs. In fact, the cycle rounding algorithm does not use the cost data when

building the maintenance schedule. Rather, it exploits the tree structure to produce a high quality solution. The shifted power-of-two algorithm computes a number of policies that maintain each component at a frequency that is a power of two times a specified base (or shifting parameter). We identify a small class of shifted power-of-two policies (of size less than or equal to the number of components) that depend only on the cycle limits and not on the cost parameters. Moreover, the best of these policies is guaranteed to have a cost within $1/\ln(2)$ of the cost of an optimal policy.

While we first develop both algorithms based on an infinite horizon, in Chapter 3 we study two finite horizon integer programming formulations, bound their integrality gaps, and show that the worst case guarantee for the cycle rounding algorithm also holds for finite horizon instances. Also in Chapter 3, we utilize a linear program to extend the shifted power-of-two algorithm to the case in which the cycle limits are initially unknown but are revealed prior to implementation. With a sufficiently large number of *rounding parameters*, this a-priori extension yields the same approximation guarantee as the original shifted power-of-two algorithm. These efficient algorithms are first the of their kind to address preventative modular maintenance scheduling while also addressing many of the underlying qualities needed for operational implementation.

The models explored in Chapters 2 and 3 only consider a single modular system. However, the an operational setting there will be many modular systems to manage that share resources including maintenance capacity, module inventory, and transportation. In Chapter 4 we develop a detailed mathematical model of module maintenance, system assembly, and transhipment in support of operations at numerous locations. To address the operations phase, we develop the *Modular Maintenance and System Assembly Model* to balance the tradeoffs between inventory, maintenance capacity, transportation resources, and operations (see Figure 1-2). In particular, this model seeks to meet external demand for serviceable modular systems with limited maintenance, transportation, and inventory resources. In addition, this model explicitly captures the risk-pooling effects of a central repair facility. Due to the complex interactions between system assembly, module repair, and transportation, the full

model is intractable for all but the smallest instances.

Due to the complexity of the model, in Chapter 5 we develop a hierarchical decomposition strategy to efficiently solve the problem. We first solve a relaxed version of the full model and then decompose the problem into two parts, the *system assembly* portion and *module repair* portion. The roots of the decomposition strategy are firmly planted in the operational realities faced by the maintenance community. In particular, the disparity between the lengths of time required to perform different types of maintenance and system assembly. By using inventory information from a relaxed version of the full model, the system assembly and module repair submodels can be solved over relatively short time horizons without significantly impacting their long term performance. Also, by using the same underlying mathematical model, for both the planning and execution phases, policy makers and frontline maintainers have a common baseline to work from. This stands in contrast to many planning models which focus on cost rather than direct support of operations. In addition, our decomposition approach allows decision makers from different organization to understand the impacts of their decision on other organizations.

Despite our initial decomposition, the system assembly model remains complex and we develop multiple algorithms to efficiently solve the problem based on different relaxations. Using the decomposition, the module maintenance problems are small enough to be solved via commercial software. Finally, we tie together the Modular Maintenance and System Assembly Model with key concepts from the Modular Maintenance Scheduling Problem to propose an integrated methodology for design and operation.

While the decomposition strategy significantly reduces the size of the resulting models, the solution time for the system assembly model remains sufficiently high to prevent quick turn analysis useful in planning and scheduling. Accordingly, by making three different assumptions about the inventory, maintenance capacity, and objective function, we develop a network flow based algorithm and two greedy algorithms that provide high quality solutions within a few seconds. As part of the module repair problem, which is small enough to be solved via commercial integer programming

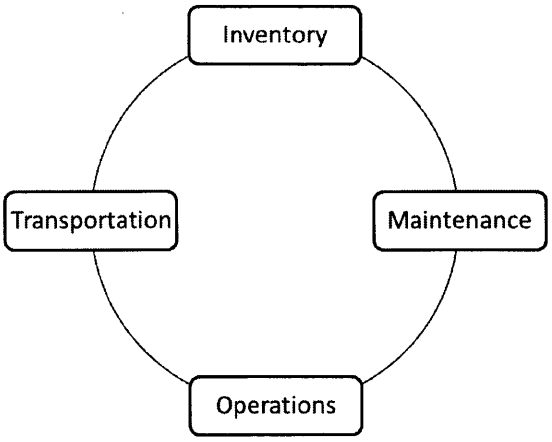

Figure 1-2: Tradeoffs

software, we explore stochastic degradation of modules. For instance, modules that are placed in service may return to the repair facility with higher levels of degradation than originally expected. This degradation may result from increased system usage, harsher operations conditions, or random events that damage the component. As part of the module repair problem, we show how such degradation can be considered in the optimization problem as well as a probabilistic analysis that can be used to help set near term maintenance manpower requirements. Finally, we exploit the idea of nestedness first introduced in Chapter 2 to develop an efficient methodology that ties together system design and operation.

Chapter 2

Modular Maintenance Scheduling

In this chapter, we study new models for scheduled preventative maintenance for modular systems that consist of multiple components each with respective operational *cycle limits*. The cycle limit for each component specifies the time interval in which this component must be repaired or replaced. The goal is to compute a feasible maintenance schedule that minimizes the cost associated with component maintenance. The typical cost structures that arise in practical settings are complex because the cost to repair several components at once is typically less than the cost (and/or manpower) required to repair each component individually. As a result, it is often cost effective to maintain components earlier than required, in conjunction with other components, to achieve cost savings. Deciding when to combine preventative maintenance actions makes the resulting models computationally challenging.

We develop two efficient and operationally tenable approximation algorithms. Both algorithms capture the implicit requirement for simple maintenance schedules. We prove tight constant factor worst-case guarantees for both algorithms showing that restricting our focus to the maintenance policies we consider does not have a large impact on the cost of a maintenance schedule. Finally, we present computational results that show these algorithms perform within a few percent of optimality on operationally relevant instances. Applications of these models arise in Air Force aircraft maintenance as well as other arenas with required preventative maintenance.

2.1 Introduction

We consider the management of scheduled maintenance activities for modular systems, such as an aircraft engine, in which the components of the system have cycle limits that specify the maximum number of periods of use between subsequent maintenance actions. For example, a cycle for the starter system in an aircraft engine could be one startup sequence. For components in an aircraft braking system, a cycle could be one landing sequence. Each component can be used for a certain number of cycles and then must be repaired or replaced due to safety or failure concerns. These cycle limits are determined through a number of methods including physical testing, simulation, and analytical assessment. Although it is possible that components fail prior to their cycle limits, due to the conservative nature of these cycle limits, such events are extremely rare. As a result, it is common to assume that a component is operational until its cycle limit is reached and that after maintenance it again has a full cycle limit.

While the systems we consider operate in continuous time, a sortie or day of usage will consume a given number of cycles. In most cases, the rate by which cycles are used has very little variability. Accordingly, we assume, by normalization if necessary, that the cycle limits are all integer multiples of a given time epoch, which we call a period. Without loss of generality, we assume that after normalization one cycle occurs in each period. Such an assumption is reasonable for many scheduled maintenance activities including Air Force maintenance. The goal in the models we develop is to coordinate a sequence of scheduled maintenance activities over a planning horizon of finitely, or infinitely, many discrete periods. In particular, the goal is to find a feasible maintenance policy in which components are maintained within their respective cycle limits and that has the minimum total, or long run average cost, respectively.

A general, and realistic, way to model the maintenance costs of the components is through cost functions that are nonnegative, increasing, and *submodular* in the subset of components being maintained. However, finding the optimal policy under these assumptions is computationally challenging. Moreover, the optimal policy can

be very complex and does not provide the operational simplicity that is essential for implementation in many practical settings. A natural approach that is often considered in practice is to use *frequency-based* policies that maintain each component at a fixed frequency. The question that arises is the increase in cost associated with using potentially suboptimal frequency-based policies. In Section 2.4 we develop two approximation algorithms that compute frequency-based policies. These algorithms provide robust and easily implemented solutions that meet the operational considerations of the maintenance community. In Section 2.4.2 we consider tree-based submodular cost functions. For this case, we develop an algorithm that is called *cycle rounding* in which the maintenance frequencies of the various components are computed by iteratively rounding the respective cycle limits. This algorithm has a worst-case performance guarantee of 2 for finite and infinite horizon instances. That is, for any instance of the problem, the solution provided by the algorithm is guaranteed to have a cost at most twice the cost of an optimal policy, which is not necessarily frequency based. In Section 2.4.3 we consider models with general submodular cost functions and develop a second algorithm, that computes *shifted power-of-two* policies that maintain each component at a frequency that is a power of two times a specified base (or shifting parameter). We identify a small class of shifted power-of-two policies (of size less than or equal to the number of components) that depend only on the cycle limits and not on the cost parameters. Moreover, the best of these policies is guaranteed to have a cost within $1/\ln(2) \approx 1.4427$ of the cost of an optimal policy. This is in fact true for *all* possible increasing submodular cost functions. In extensive computational experiments, the cycle rounding and shifted power-of-two policies perform within a few percent of optimal. As a byproduct, this shows that frequency-based policies are indeed near-optimal.

Many modern systems consist of multiple components arranged in a modular design. Performing maintenance on (or replacing) a component requires removal of the module containing the component. Once the component is repaired and returned to a serviceable condition, the repaired component and module is reinserted into the system. In many cases, the system design dictates that the removal of one module

requires removal of other modules. For instance, the F100 engine, a US Air Force engine used in fighter aircraft, is composed of five major modules: the fan module, the core module, the fan drive turbine module, the augmentor/exhaust module, and the gearbox module. If maintenance is required on a component in the fan drive turbine module, the augmentor/exhaust module must be removed to access the fan drive turbine module [21]. Such maintenance actions are costly and time consuming due to the required teardown, maintenance, reassembly, and testing. For the F100 engine this process can take 12 to 21 days without considering inventory or manpower delays [7]. For an aircraft engine, frequent or extended system downtimes have a negative effect on mission capability rates for the corresponding aircraft. In addition, maintenance activities can require specialized manpower and equipment which are limited.

Submodular cost structures arise in several specific maintenance contexts in the U.S. Air Force. One way to model the maintenance costs in modular systems is through an out-rooted directed tree. Figure 2-1 shows a partial example for the F100 engine. Directed arcs in the tree correspond to the modular dependencies that exist between the connected nodes (i.e., the parent node must be removed in order to repair any of its child nodes). A *dependency path* from the root to a leaf describes the order in which modules must be removed and replaced to maintain a specific component. Each node has an associated cost that reflects the time, manpower, and/or inventory utilization required to perform the specific maintenance activity. To perform maintenance on a component, all modules on its dependency path must be removed in the order specified by the path. The cost of a maintenance activity for a component is the sum of the costs of all nodes on the dependency path for that component. Once all parent modules have been removed and the maintenance on the component is completed, the modules are reassembled in the reverse order of the dependency path. Due to the modular construction, we assume, without loss of generality, that maintenance is required only on leaves of the dependency tree, which correspond to the cycle-limited components. These leaves might, however, be at different depths in the tree. As we note later, the resulting cost function is

submodular.

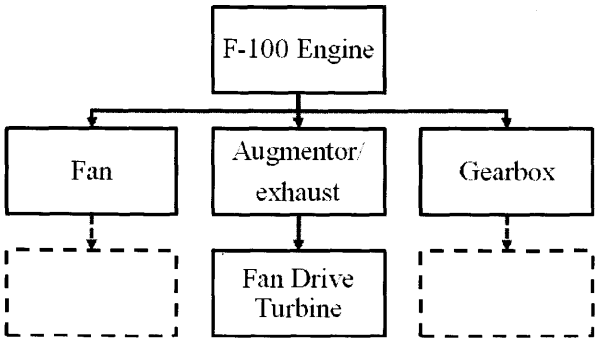

Figure 2-1: Partial dependency tree for the F100 engine.

Another submodular cost function of interest arises in situations where the cost incurred by maintaining a subset of components in a given time period is the maximum cost of all the component included in the subset. Returning to the earlier tree example, if the node costs represent time, the time required for a maintenance action is computed by considering the length of the longest path between the root and each of the of the leaves (components) included in the maintenance action. In particular, the length of time required for the maintenance action is equal to the length of the longest path. This cost function implicitly assumes that maintenance occurs in parallel for unconnected modules and components.

Next we discuss several practical settings within the U.S. Air Force in which the models and algorithms studied in this chapter could be of use. During the design and testing phase of a modular system, a development team could use the models and algorithms we discuss to explore the short and long term implications of changing various design parameters such as early investment in maintenance capability to reduce per incident maintenance costs or increasing component cycle limits to decrease the future sustainment costs for the system. Once a system has been fielded, operational considerations or challenges might affect the maintenance philosophy codified during the development phase. During this operational phase, actual maintenance costs or costs from an auxiliary optimization problem (i.e., linear programming dual variables) could be used to produce feasible and low cost maintenance schedules. The models

and algorithms studied in this chapter could also be used as building blocks within more comprehensive operational maintenance and inventory management models.

We will review the relevant maintenance scheduling literature in Section 2.2 and then will define the Modular Maintenance Scheduling Problem in Section 2.3. In Section 2.4 we will develop a lower bound on any solution to the Modular Maintenance Scheduling Problem and use this lower bound to prove worst case performance guarantees for two algorithms. Finally, extensive computational results in Section 2.5 show that these approximation algorithms perform much better than their respective approximation guarantees, within a few percent of optimal for many realistic instances.

2.2 Literature Review

Numerous researchers have examined maintenance models for multi-component systems with economic and/or structural dependencies. For situations with economic dependencies, the cost of maintaining a set of components does not equal the sum of individual maintenance costs for these components. For situations with structural dependencies, components form a module which forces the maintenance of several components together. The existing literature in this field is extensive and it is beyond the scope of this chapter to discuss all the relevant research contributions; instead, we refer the reader to related surveys [27, 49] for a comprehensive review of the literature. Most of the literature focuses on infinite horizon problems with stochastically failing components and attempts to find optimal maintenance schedules that minimize either downtime or maintenance costs. Two widely studied models in the maintenance literature are the k-out-of-n model and a model with n constant failure rate (CFR) components and one increasing failure rate (IFR) component. The first model assumes the system has n identical components of which k must be functional for the system to operate. Most of the work in this area has focused upon $k < n$ and on developing policies that balance the downtime and preventative maintenance costs. In the second model, the decision to be made is when to perform preventative

maintenance on the single IFR component either when a CFR component fails or when the IFR component has reached a certain time in service. In this model, the goal is to minimize maintenance costs, which might differ depending on the state of the IFR component when maintenance is initiated.

In contrast, despite its operational relevance, very little progress has been made for more general settings, such as the maintenance of multi-component systems with multiple setup activities. In fact, Kobbacy et al. [27] specifically state in reference to models that consider multiple setup activities over a finite horizon "We have found one article in this category... this is the first attempt to model a maintenance problem with a hierarchical set-up structure." van Dijkhuizen [47] studied the problem of grouping preventive maintenance actions in a multi-step, multi-component production system. He models the multi-component production system as a tree with leaves representing components that require maintenance within specified frequencies. The model he develops assumes that maintenance actions can occur only at integer multiples of a selected, part-dependent, preventative maintenance frequency. Under this assumption, the preventative maintenance schedule repeats itself over the finite planning horizon. An integer programming formulation is then used to find the best cyclic policy. van Dijkhuizen and van Harten [48] proposed a combinatorial algorithm for the same model. For instances with a single common set-up activity (i.e., the underlying tree is a star), they developed a polynomial time dynamic programming algorithm to calculate the optimal solution. For instances with multiple shared set-up activities, they develop a branch-and-bound algorithm that terminates with the optimal policy but is not polynomial. Neither of these algorithms address the additional cost incurred by restricting attention to policies that are frequency-based versus more general non-cyclic policies.

The models and the algorithms studied in this thesis have some similarities with inventory models that have been studied in the 1970's through the 1990s. With that said, there are several fundamental difference between the maintenance models we study and these inventory models. Moreover, the way our algorithms are devised as well as the analysis that we obtain are entirely different then the body of work on

these inventory models.

The complexity of continuous time infinite horizon inventory models with stationary deterministic demand rates in a multi-stage production/inventory environment has prompted the development of specialized algorithms that focus on imposing specific structure on the solution, specifically stationary-nested policies, with varying performance [14]. The effectiveness of power-of-two policies were first established in the seminal papers of Roundy [40] and Maxwell and Muckstadt [30]. They showed that power-of-two policies are within 1.06 of optimal if one does not optimize the base (shift parameter) and within 1.02 if the shift parameters is optimized. The analysis is based on a nonlinear relaxation of the problem and the policies tightly depend on the respective cost parameters.

In contrast to these models, our models have discrete time periods and allow either a finite or infinite horizon. In addition, our models have a significantly more complex ordering (maintenance) cost structure compared with the graph-based additive cost structure in the inventory models. Even if one considers a special case of our model where the costs are defined on an acyclic graph, whenever a leaf orders the entire path to the root has to order, whereas in the inventory models this is not necessary. In addition, in our models there is a capacity constraint that enforces upper bounds on the time between two consecutive orders (maintenance actions).

Federgruen and Zheng [20] studied the lot-sizing/inventory models above but with a similar upper bound constraint on the size of the reorder interval and showed, using very similar techniques to the initial work of Roundy [40] that power-of-two policies are again within 1.06 of optimal.

With an additive cost structure, Roundy [41] studied the above inventory models with a lower bound capacity constraint that restricts the reorder interval from being too small. Interestingly, he showed that power-of-two policies are within $1/\ln(2)$ of optimal, obtaining a similar worst-case guarantee as we obtained for the maintenance models that have general ordering (maintenance) cost structure and upper bounds on the reorder interval. The analysis Roundy used is similar to in his initial work using the same type of nonlinear relaxations.

Federgruen et al. [19] considered a more general cost structure where the ordering cost is a general submodular function in the subset of items being ordered, but with no capacity constraints. They are able to show that a power-of-two policies are within 1.02 of optimal. More recently, Teo and Bertsimas [45] consider similar models and replicated the above results with a few generalizations by applying randomized algorithms to the optimal solution of the nonlinear relaxation used in the previous works, as well as improved relaxations.

To summarize, none of the existing results apply to the models that we studied in this work (i.e., with general submodular cost functions and upper bound capacity constraints). Moreover, all of the existing work employs non-linear relaxations that depend on the cost parameters. In contrast, we obtain robust policies that can be computed with minimal or no dependence on the costs parameters. In addition, our analysis is based on different lower bounds and randomized techniques.

2.3 The Modular Maintenance Scheduling Problem

Consider a system with a set of components, denoted by $\mathcal{C}$, that must be maintained over a discrete-time planning horizon of either T or infinite number of periods. Each component i has a component-specific cycle limit f_i that specifies the maximum allowable number of periods between two consecutive maintenance actions of this component. Each time a component is repaired/replaced, it starts with a full cycle limit. The goal is to compute a feasible maintenance schedule that minimizes the total cost for finite horizon problems, or long run average cost for infinite horizon problems. These costs capture the time and/or expense associated with the corresponding component maintenance actions.

We assume that all components have a full cycle life at the beginning of the first period in the planning horizon. At the beginning of each period, a decision is made about which components, if any, should be maintained in that period. Our

assumption is that maintenance occurs instantaneously and the associated costs are incurred. After any maintenance actions have been carried out, the system is used for one period and the remaining cycles for all components are decremented by one. The assumption that maintenance occurs instantaneously follows directly from the assumption that the system requires all components to be functional. The time associated with maintenance actions is implicitly captured in the cost structure.

We assume without loss of generality that $1 < f_i < T$ and that the components are numbered in nondecreasing order of their cycle limits (i.e., $f_i \leq f_{i+1}$). If $f_i = 1$ for some component i, then component i can be scheduled for maintenance in every time period over the planning horizon and removed from the problem. If $f_i \geq T$ for some component i, then component i does not need to be maintained during the planning horizon and can be discarded from the problem. For each subset of components $S \subseteq \mathcal{C}$, let $K(S)$ be the total cost of maintaining these components in a given time period. We assume $K(\cdot)$ is a nondecreasing submodular (see definition below) set cost function defined, for which, without loss of generality $K(\emptyset) = 0$.

Definition 2.3.1 (Submodular function) *A real-valued function defined on subsets of a ground set $\mathcal{C}$ is submodular if for two subsets, S and S', of $\mathcal{C}$,*

$$K(S) + K(S') \geq K(S \cup S') + K(S \cap S').$$

An equivalent definition of a submodular function focuses on economies of scale. A real-valued function defined on subsets of a ground set $\mathcal{C}$ is submodular if for every $S \subset S' \subset \mathcal{C}$ and $i \in \mathcal{C} \setminus S'$:

$$K(S \cup \{i\}) - K(S) \geq K(S' \cup \{i\}) - K(S').$$

A feasible solution to the problem defines a maintenance schedule in which component maintenance actions are no more than f_i periods apart. The cost of the solution is the sum of the maintenance costs incurred in each period over the planning horizon,

or the long run average cost in infinite horizon models.

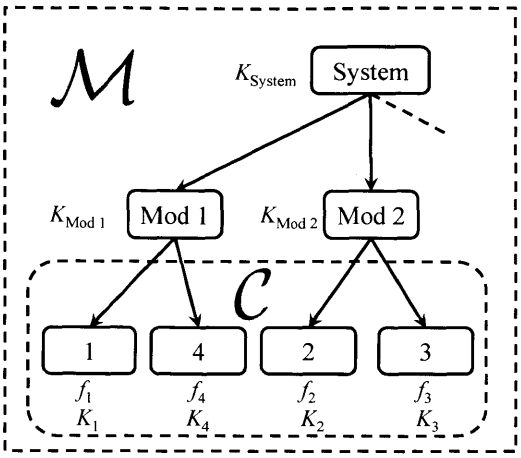

Figure 2-2: A schematic illustration of $\mathcal{C}$, $\mathcal{M}$, cycle limits, and maintenance costs for a system modeled by a tree.

One very common submodular cost function can be defined on an out-rooted tree where the root corresponds to the entire system, the leaves correspond to components, and the other nodes in the tree correspond to modules. Let $\mathcal{M}$ denote the set of all modules and all components (leaves), including the entire system and let P_i denote the *dependency path* for component i. That is, the path from the root to (and including) a leaf that describes the order in which modules must be removed and replaced to maintain a specific component. Specifically, a module j belongs to P_i if and only if module j must be removed to perform maintenance on component i. Let $K_j \geq 0$ denote the cost to remove and replace module $j \in \mathcal{M} \setminus \mathcal{C}$ or to repair component $j \in \mathcal{C}$. In this case, the cost of maintaining a set S of components is equal to the sum of module removal/replacement and component maintenance costs over all modules and components that reside in the union of the dependency paths $\{P_i : i \in S\}$, that is,

$$K(S) = \sum_{j \in \bigcup_{i \in S} P_i} K_j.$$

It is easy to see that this specialized cost structure is submodular, nonnegative, and $K(\emptyset) = 0$. Figure 2-2 illustrates the sets $\mathcal{C}$ and $\mathcal{M}$, cycle limits, and maintenance costs for a small dependency tree and Figure 2-3 shows the dependency path for

component 1. A more general version of this model could include systems that can be represented by a directed, acyclic graph in which a module might depend upon multiple, higher level modules.

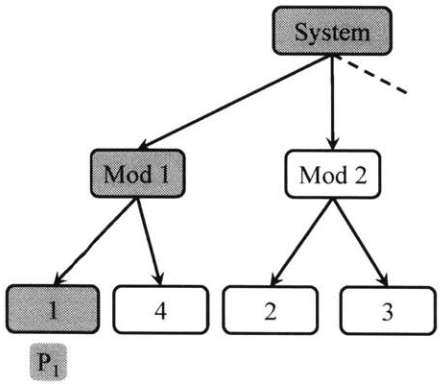

Figure 2-3: Dependency path for component 1.

A second example of a submodular cost function of particular interest is the overall downtime cost. This could be defined on a tree. The residual path P_i will be the same as defined above. The parameters K_j will be the time required to remove and replace module $j \in \mathcal{M} \setminus \mathcal{C}$ or to repair component $j \in \mathcal{C}$. The cost of maintaining a subset of components S is equivalent to the longest path from the root to some component $i \in S$. That is,

$$K(S) = \max_{i \in S} \sum_{j \in P_i} K_j.$$

Again it can be verified that $K(S)$ above is indeed submodular.

2.4 Approximation Algorithms

In this section, we develop several frequency-based (cyclic) policies that are easy to compute, and show that they are provably near-optimal. That is, the cost of these policies is guaranteed to be within a constant factor of the optimal cost, uniformly for all problem instances. Note again that the optimal cost policy might not be frequency-based and thus hard to compute and implement in practice. We start by developing general lower bounds on the cost of any feasible policy (including the optimal policy)

and then use these lower bounds to prove the approximation guarantees for the two algorithms.

2.4.1 Lower Bound

To develop the lower bounds, we introduce a *charging scheme* that is used to allocate the total cost of each maintenance action among all of the components included in that action. This leads to a decomposition of the total cost of any policy by allocating it to components. For a maintenance action including the set of components $S \subseteq \mathcal{C}$ and each $i \in S$, we let $S_i = S \cap \{1, 2, \ldots, i-1\}$ be the subset of lower indexed components, those components with a cycle limit that is lower than or equal to the cycle limit of component i, that are included in the maintenance action. We define $S_i = \{\emptyset\}$ if i is the lowest indexed node in the set S. Component i is then charged with the marginal additional cost of adding component i to a maintenance action including only S_i. That is, it is charged $K(S_i \cup \{i\}) - K(S_i)$. Since $K(\cdot)$ is nondecreasing, the charge to each component is nonnegative and the full cost of the maintenance action is charged to components as

$$\sum_{i \in S} (K(S_i \cup \{i\}) - K(S_i)) = K(S).$$

We also define the *residual cost* for a component i, denoted by K^i, to be $K(\{1, 2, \ldots, i\})$-$K(\{1, 2, \ldots, i-1\})$. Note that since $K(\cdot)$ is submodular, this is the minimum amount component i can be charged for any maintenance action, regardless of the other components maintained in conjunction with component i. In addition, the submodularity of $K(\cdot)$ implies that there exists a component $m(i) \leq i-1$ for which

$$K^i = K(\{1, 2, \ldots, m(i)\} \cup \{i\}) - K(\{1, 2, \ldots, m(i)\}). \tag{2.1}$$

That is, for all $m(i) \leq j \leq i-1$, $K(S_j \cup \{i\}) - K(S_j) = K^i$. We refer to $m(i)$ as the *parent* of i. For example, in Figure 2-3 above, $m(4) = 1$.

Using this charging scheme, we can develop lower bounds on the optimal solution

for both finite and infinite horizon problems. For both lower bounds, we will use the fact that the residual cost, K^i, is the minimum amount a component can be charged when it is maintained and that every feasible schedule must maintain each component i at least once each f_i periods.

Lemma 2.4.1 *The cost of an optimal solution, denoted by* OPT*, is bounded from below as follows:*

$$\sum_{i \in \mathcal{C}} \left\lfloor \frac{T}{f_i} \right\rfloor \cdot K^i \leq \text{OPT} \qquad \text{Finite horizon,}$$

$$\sum_{i \in \mathcal{C}} \frac{K^i}{f_i} \leq \text{OPT} \qquad \text{Infinite horizon.}$$

Proof : Each component i must be maintained at least once during each of the $\lfloor T/f_i \rfloor$ disjoint intervals of length f_i, incurring the minimum charge K^i per maintenance action, yielding the finite horizon result. Dividing by T and taking the limit as T goes to infinity yields the infinite horizon result. ■

Note that the lower bounds in Lemma 2.4.1 are also valid for continuous time models in which a maintenance action can take place at any point in time. Accordingly, the approximation results shown later in Sections 2.4.2 and 2.4.3 are valid even when we allow maintenance actions to occur at non-integral periods. From an operational standpoint, this allows us to consider relatively coarse time periods without significantly impacting the quality of the resulting maintenance schedule.

Relating to the charging scheme described above, we now define a *nested schedule* to be one in which each component i is maintained only if it is charged exactly K^i. That is, for every component i included in a maintenance action on a subset of components $S \subset \mathcal{C}$

$$K(S_i \cup i) - K(S_i) = K^i.$$

Following the discussion above, we consider schedules in which component i is maintained only if its parent $m(i)$ is also maintained in the same time period, implying that component i will be charged exactly K^i.

As shown next, the infinite horizon lower bound will permit us to obtain worst-case performance guarantees for nested and frequency-based (cyclic) schedules by bounding the maximum ratio over all components between the frequency that component i is maintained and the minimum frequency, $1/f_i$.

Theorem 2.4.2 *Consider a frequency-based and nested policy with maintenance cycles $\widehat{f}_i \leq f_i$ for each $i \in \mathcal{C}$. For an infinite horizon model, the long run average cost of this schedule is at most $\max_{i\in\mathcal{C}}(f_i/\widehat{f}_i)$ times the optimal long run average cost, denoted by* OPT.

Proof : The maintenance frequency for each component i is $1/\widehat{f}_i$. Since the schedule is nested, component i is charged exactly K^i per maintenance action and thus contributes $K^i/\widehat{f}_i$ to the long run average cost. Thus, the total long run average cost of the schedule is

$$\sum_{i\in\mathcal{C}} \frac{K^i}{\widehat{f}_i}.$$

This implies:

$$\text{OPT} \leq \sum_{i\in\mathcal{C}} \frac{K^i}{\widehat{f}_i} = \sum_{i\in\mathcal{C}} \frac{f_i \cdot K^i}{f_i \cdot \widehat{f}_i} \leq \max_{i'\in\mathcal{C}} \frac{f_{i'}}{\widehat{f}_{i'}} \cdot \sum_{i\in\mathcal{C}} \frac{K^i}{f_i} \leq \max_{i'\in\mathcal{C}} \frac{f_{i'}}{\widehat{f}_{i'}} \cdot \text{OPT}$$

with the final inequality resulting from the lower bound on OPT, obtained in Lemma 2.4.1. ∎

Note that the worst-case performance guarantee in Theorem 2.4.2 is based solely on the scheduled maintenance frequencies and not affected by the cost parameters. Accordingly, we develop two approximation algorithms that compute cyclic and nested policies. We describe these algorithms assuming that time is continuous and thus the resulting maintenance schedules can be fractional. In the problem formulation, maintenance actions are scheduled only at discrete time periods. However, we can convert the resulting continuous time (fractional) maintenance schedule into a discrete time maintenance schedule with no increase in cost. Consider a cyclic and nested continuous time schedule with maintenance actions scheduled for component i

every $\widehat{f}_i \in \mathbb{R}^+$ time periods. The corresponding discrete time schedule for component i will be

$$\left\lceil \widehat{f}_i \right\rceil, \left\lceil 2 \cdot \widehat{f}_i \right\rceil, \left\lceil 3 \cdot \widehat{f}_i \right\rceil, \ldots.$$

This rounded schedule is feasible because the time between consecutive maintenance actions is at most f_i. Specifically,

$$\left\lceil (k+1) \cdot \widehat{f}_i \right\rceil - \left\lceil k \cdot \widehat{f}_i \right\rceil \leq \left\lceil k \cdot \widehat{f}_i + f_i \right\rceil - \left\lceil k \cdot \widehat{f}_i \right\rceil = \left\lceil k \cdot \widehat{f}_i \right\rceil + f_i - \left\lceil k \cdot \widehat{f}_i \right\rceil = f_i$$

The first inequality holds because $\widehat{f}_i \leq f_i$ and the first equality follows from the integrality of f_i. Since the fractional schedule was nested, the new discrete schedule is also nested. Consequently, at every maintenance action that includes component i, it will be charged only its residual cost. In addition, the rounding up does not increase the number of maintenance actions for each component i. Therefore, the cost of the new solution does not increase.

2.4.2 Cycle Rounding Algorithm

Next we focus on the tree-based model described in Section 2.3. We describe a simple algorithm that we call *cycle-rounding*, which schedules maintenance actions iteratively, starting with component 1. As we shall show, the algorithm computes a nested, cyclic policy with a total cost at most twice the optimal cost. Recall that the components are numbered in nondecreasing order of their cycle limits (i.e., $f_i \leq f_{i+1}$), and that P_i denotes the dependency path for component i (i.e., the path from the root of the tree to component i).

To describe the algorithm, define the *residual path* R_i of component $i \in \mathcal{C}$ to be component i and all modules that are on the dependency path of i but not on the dependency paths for any lower indexed components $\{1, 2, \ldots, i-1\}$. That is, $R_i = P_i \setminus \bigcup_{i'<i} P_{i'}$. Following the discussion above, observe that from the definition of R_i, we have $R_i = P_i \setminus P_{m(i)}$, where $m(i)$ is defined as before (see the discussion of Equation (2.1)).

Let f_i^{CR} denote the respective maintenance cycles of the cycle rounding algorithm schedule. That is, component i will be maintained every $f_i^{\mathrm{CR}} \leq f_i$ periods. Next we describe how to iteratively set the values of f_i^{CR} for all $i \in \mathcal{C}$. For $i = 1$, set $f_1^{\mathrm{CR}} = f_1$. For all components $i > 1$ in order, consider $f_{m(i)}^{\mathrm{CR}}$ (previously computed since $m(i) < i$) and set f_i^{CR} to be the largest multiple of $f_{m(i)}^{\mathrm{CR}}$ that is still smaller than f_i. That is

$$f_i^{\mathrm{CR}} = \left\lfloor \frac{f_i}{f_{m(i)}^{\mathrm{CR}}} \right\rfloor \cdot f_{m(i)}^{\mathrm{CR}} \leq f_i.$$

Observe that for the tree-based model, the residual cost K^i is exactly $K^i = \sum_{j \in R_i} K_j$. Moreover, by construction, the cycle rounding algorithm schedules maintenance actions in periods in which component $m(i)$ is also maintained. Therefore, the cycle rounding algorithm generates a nested, cyclic schedule.

We next show that the cycle rounding algorithm is a 2-approximation algorithm. That is, the cost of the schedule produced by the cycle rounding algorithm is at most twice the optimal cost (considering any feasible policy).

Theorem 2.4.3 *The cycle rounding algorithm guarantees an approximation ratio of 2 for the modular maintenance scheduling problem.*

Proof : Since the schedule generated by the cycle rounding algorithm is cyclic and nested, by Theorem 2.4.2 it suffices to show that $f_i / f_i^{\mathrm{CR}} \leq 2$ for all $i \in \mathcal{C}$. However, this follows directly from the fact that by construction

$$f_i - f_i^{\mathrm{CR}} = f_i - \left\lfloor \frac{f_i}{f_{m(i)}^{\mathrm{CR}}} \right\rfloor \cdot f_{m(i)}^{\mathrm{CR}} \leq f_{m(i)}^{\mathrm{CR}} \leq f_i^{\mathrm{CR}}.$$

■

Observe that the cycle rounding algorithm produces a near-optimal schedule that depends only on the cycle limits and the tree structure, but not on the maintenance costs. This property is attractive for practical algorithms since the costs associated with maintenance can be hard to define, difficult to estimate accurately, and/or change over time. In particular, the maintenance costs might include man hours,

use of specialized tooling, holding inventory, and the use of specialized maintenance facilities. Since it does not depend upon maintenance costs and has well-defined, intuitive scheduling rules, this algorithm could be implemented in a maintenance setting without requiring frequent updates or the execution of complex operational plans.

Another benefit of the cycle rounding algorithm is its robustness to changes in the component cycle limits. The cycle limit for component i can be reduced by $f_i - f_i^{\text{CR}}$ and increased by $f_{m(i)}^{\text{CR}} - (f_i \bmod f_{m(i)}^{\text{CR}})$ without changing the resulting schedule. Intuitively, the allowable decrease is based on the feasibility of f_i^{CR} and the increase upon $f_i - \lfloor f_i / f_{m(i)}^{\text{CR}} \rfloor \cdot f_{m(i)}^{\text{CR}}$ remaining less than $f_{m(i)}^{\text{CR}}$. While this robustness toward changes in cycle limits might not be useful in an operational setting, during the system's design and testing phase it can prove valuable. Designers will be able to tailor margins of safety and engineers will be able to develop appropriate test plans to set the cycle limit of a component appropriately considering the tradeoff between near term investments and longer term maintenance costs.

The cycle rounding algorithm can also be applied to the downtime cost function in which the cost incurred for any maintenance action is the maximum cost among all components included in the maintenance action. For component i, $m(i)$ will be the component for which $[K(i) - K(m(i))]^+ = K^i$ where $K(i)$, as defined earlier, is the cost of maintaining component i. If there are multiple such components, we will choose $m(i)$ to be the component which yields rounded cycle limit closest to f_i. In particular $m(i)$ will be the component for which $[K(i) - K(m(i))]^+ = K^i$ and $f_i - \lfloor f_i / f_{m(i)}^{\text{CR}} \rfloor \cdot f_{m(i)}^{\text{CR}}$ is minimal. Thus, $m(i)$ can be uniquely defined. By the previous analysis, the resulting schedule is cyclic, nested, and has a maximum maintenance frequency increase of 2. This implies that the cycle rounding algorithm is a 2-approximation for the downtime cost function. However, its reliance on the tree structure or the ability to identify a single parent $m(i)$ for each component, precludes using the algorithm for more general submodular cost functions. One such submodular cost function that can not be addressed with the cycle rounding algorithm is additive costs on a the directed, acyclic graph (instead of a tree). This cost function could arise if two or more modules need to be removed to access a lower level module

or component.

Cycle Rounding Algorithm – Bad Example

Consider an instance of the modular maintenance scheduling problem in which there are two components and one module. The module maintenance cost is ε as is the maintenance cost for component 1. However, the maintenance cost for component 2 is 1. The cycle limit for component 1 is 2^κ and the cycle limit for component 2 is $2^{\kappa+1} - 1$, for some positive integer κ. Figure 2-4 illustrates this example.

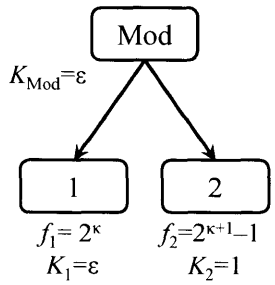

Figure 2-4: Cycle rounding algorithm bad example.

For ε sufficiently small, an optimal policy will schedule component 1 every 2^κ periods and component 2 every $2^{\kappa+1} - 1$ periods. Accordingly, the optimal schedule will have a long run average cost of

$$\frac{2 \cdot \varepsilon}{2^\kappa} + \frac{1 + \varepsilon}{2^{\kappa+1} - 1} - \frac{\varepsilon}{2^\kappa \cdot (2^{\kappa+1} - 1)}.$$

The cycle rounding algorithm will schedule both components every 2^κ periods and will incur $1 + 2 \cdot \varepsilon$ units of cost every 2^κ periods, which yields a long run average cost of

$$\frac{1 + 2 \cdot \varepsilon}{2^\kappa}.$$

As $\varepsilon \to 0$, the ratio of these average period costs converges to $(2^{\kappa+1} - 1)/2^\kappa = 2 - 1/2^\kappa$. As $\kappa \to \infty$, the ratio converges to 2 which implies the preceding analysis is tight for the cycle rounding algorithm under the general assumption of nonnegative costs. This tight example is, however, very pathological and unlikely to arise in practice.

2.4.3 Shifted Power-of-Two Algorithm

While the cycle rounding algorithm is an efficient approximation algorithm that yields nested, cyclic schedules, it is not well defined for more general submodular cost functions. In addition, there might be operational considerations that necessitate a more restricted cyclic maintenance schedule in which a system is repaired only in a *fully nested* manner, in which component i is maintained only if components $\{1, 2, \ldots, i-1\}$ are also maintained. Clearly, any fully nested schedule is also nested. Such considerations might include equipment availability, manpower scheduling, and inventory management.

Building on the worst case instance described in Section 2.4.2, we consider a more refined rounding scheme. The key insight from the cycle rounding worst case instance and Theorem 2.4.2 is that the performance guarantee is strongly affected by the amount a cycle limit is rounded down from f_i. If, by shifting the rounding point, we can reduce the overall amount of rounding that occurs, we might improve the quality of the solution. We first consider a continuous time relaxation of the problem in which maintenance actions can be scheduled in fractional time periods. Following the discussion in Section 2.4.1, we will then round the solution to get a feasible, discrete solution with no higher cost.

We consider a subset of frequency-based policies called *shifted power-of-two* policies in which the rounded cycle limits for all components are powers of 2 times a base. Specifically, we parameterize this class of policies through a parameter $\delta \in [1, 2)$. For a given value of δ, the rounded cycle limits, f_i^δ, are set to be $f_i^\delta = \delta \cdot 2^\kappa$ where κ is the unique integer for which $\delta \cdot 2^\kappa \leq f_i < \delta \cdot 2^{\kappa+1}$.

For any value of $\delta \in [1, 2)$, the resulting schedule is obviously cyclic and is also fully nested because a component i is scheduled for maintenance only in periods in which components $\{1, 2, \ldots, i-1\}$ are also scheduled for maintenance. Thus, the worst-case analysis in Theorems 2.4.4 and 2.4.7 below are focused on bounding the ratio f_i/f_i^δ. Before analyzing the best shifted power-of-two policy, we develop a simpler result.

Theorem 2.4.4 *Any shifted power-of-two policy is guaranteed to have a cost that is*

at most twice the cost of the optimal solution.

Proof : Let $\delta \in [1,2)$ denote the rounding parameter for the shifted power-of-two policy. For any component i,

1. if $\delta \leq \beta_i$, then the rounded frequency for component i is $f_i^\delta = \delta \cdot 2^{\kappa_i}$, or
2. if $\delta > \beta_i$, then $f_i^\delta = \delta \cdot 2^{\kappa_i - 1}$.

In Case 1, $f_i/f_i^\delta \leq \beta_i/\delta \leq 2$ and in Case 2, $f_i/f_i^\delta \leq 2 \cdot \beta_i/\delta \leq 2$. Since the bounds are valid for all components i, Theorem 2.4.2 implies the result. ∎

The analysis in Theorem 2.4.4 assumes that δ is chosen arbitrarily. Next we discuss how δ can be chosen "optimally" and demonstrate that this leads to an improved worst-case performance guarantee. To analyze the best choice of the rounding parameter δ for the shifted power-of-two algorithm, we will first analyze an algorithm in which δ is chosen randomly from a specific distribution over the interval [1,2). This will yield an improved worst-case guarantee in expectation. We will then show how to deterministically find the optimal round parameter, δ^*, that attains this worst-case guarantee.

Theorem 2.4.5 *If U is uniformly distributed over the interval $[0, \ln(2))$ and $\delta = e^U$, the expected cost of the resulting shifted power-of-two policy is at most $1/\ln(2)$ times the cost of an optimal policy.*

Proof : Since any choice of δ yields a nested and cyclic schedule, for each component i the contribution to the long run average cost is K^i/f_i^δ. The infinite horizon lower bound developed from Lemma 2.4.1 implies the following result:

$$\frac{\mathbb{E}_\delta\left[\sum_{i\in\mathcal{C}} \frac{K^i}{f_i^\delta}\right]}{\text{OPT}} \leq \frac{\mathbb{E}_\delta\left[\sum_{i\in\mathcal{C}} \frac{K^i}{f_i^\delta}\right]}{\sum_{i\in\mathcal{C}} \frac{K^i}{f_i}} = \frac{\sum_{i\in\mathcal{C}} \mathbb{E}_\delta\left[\frac{K^i}{f_i^\delta}\right]}{\sum_{i\in\mathcal{C}} \frac{K^i}{f_i}} \leq \max_{i\in\mathcal{C}} \frac{\mathbb{E}_\delta\left[\frac{K^i}{f_i^\delta}\right]}{\frac{K^i}{f_i}} = \max_{i\in\mathcal{C}} \mathbb{E}_\delta\left[\frac{f_i}{f_i^\delta}\right].$$

The first inequality follows from the lower bound obtained in Lemma 2.4.1. The second inequality follows from the fact that we are summing over the same sets which implies the ratio of the sums is bounded by the maximum ratio of individual terms.

Now, for each $i \in \mathcal{C}$ we can write $f_i = \beta_i \cdot 2^{\kappa_i}$ for some $\beta_i \in [1,2)$ and positive integer κ_i. If $\delta \leq \beta_i$, then $f_i^\delta = \delta \cdot 2^{\kappa_i}$ and if $\delta > \beta_i$, then $f_i^\delta = \delta \cdot 2^{\kappa_i - 1}$. These cases directly relate to the situations in which the random variable $u \in (0, \ln(\beta_i)]$ and $u \in (\ln(\beta_i), \ln(2))$, respectively. In these cases we can define the ratio between f_i and f_i^δ as follows:

$$\frac{f_i}{f_i^\delta} = \begin{cases} \dfrac{\beta_i \cdot 2^{\kappa_i}}{e^u \cdot 2^{\kappa_i}} = \dfrac{\beta_i}{e^u}, & u \in (0, \ln(\beta_i)]\,; \\ \dfrac{\beta_i \cdot 2^{\kappa_i}}{e^u \cdot 2^{\kappa_i - 1}} = \dfrac{2 \cdot \beta_i}{e^u}, & u \in (\ln(\beta_i), \ln(2))\,. \end{cases}$$

If we take the expectation over U we have the following result for all components, not just the maximum one.

$$\mathbb{E}_U\left[\frac{f_i}{f_i^\delta}\right] = \int_0^{\ln(2)} \frac{f_i}{f_i^\delta} \cdot \frac{1}{\ln(2)} \cdot du = \int_0^{\ln(\beta_i)} \frac{\beta_i}{e^u} \cdot \frac{1}{\ln(2)} \cdot du + \int_{\ln(\beta_i)}^{\ln(2)} \frac{2 \cdot \beta_i}{e^u} \cdot \frac{1}{\ln(2)} \cdot du$$

$$= -\frac{\beta_i}{\ln(2)} \left(e^{-u}\Big|_0^{\ln(\beta_i)} + 2 \cdot e^{-u}\Big|_{\ln(\beta_i)}^{\ln(2)} \right) = -\frac{\beta_i}{\ln(2)} \left(\frac{1}{\beta_i} - 1 + 2 \cdot \frac{1}{2} - 2 \cdot \frac{1}{\beta_i} \right) = \frac{1}{\ln(2)}.$$

The result then follows from Theorem 2.4.2. ∎

Theorem 2.4.5 analyzes a randomized algorithm that chooses δ from a specified distribution. Lemma 2.4.6 below shows that we can in fact compute the best shifted power-of-two policy deterministically, obtaining the same worst-case guarantee of $1/\ln(2)$.

Lemma 2.4.6 *For any shifted power-of-two policy with parameter δ, there exists another shifted power-of-two policy with the same or lower cost which has a rounding parameter δ^*, satisfying $f_i^{\delta^*} = f_i$ for some component $i \in \mathcal{C}$.*

Proof : Suppose that for some rounding parameter δ for all $i \in \mathcal{C}$:

$$f_i^\delta = \delta \cdot 2^\kappa < f_i < \delta \cdot 2^{\kappa+1}.$$

By increasing δ by a sufficiently small amount, the rounded maintenance frequen-

cies will remain feasible and the time between successive maintenance actions will increase for all components, which is clearly better. Accordingly, we can increase δ until $f_i^\delta = f_i$ for some component i. ∎

For each component i, we can write $2^{\kappa_i} \leq f_i = \beta_i \cdot 2^{\kappa_i} < 2^{\kappa_i+1}$ for some number $\beta_i \in [1, 2)$ and positive integer κ_i. Lemma 2.4.6 implies that the best (optimal) shifted power-of-two policy has a rounding parameter δ^* satisfying $f_i^{\delta^*} = f_i$ for some component i. That is, $\delta^* \in \{\beta_1, \beta_2, \ldots, \beta_{|\mathcal{C}|}\}$. It follows that one can find the best shifted power-of-two policy efficiently by considering only those policies defined by these $|\mathcal{C}|$ rounding parameters. We have now obtained the following theorem.

Theorem 2.4.7 *The best shifted power-of-two policy, $\delta^* \in \{\beta_1, \beta_2, \ldots, \beta_{|\mathcal{C}|}\}$, is guaranteed to have a cost that is at most $1/\ln(2)$ times the cost of an optimal policy.*

Recall that a shifted power-of-two policy can schedule maintenance in fractional time periods. However, by applying the rounding procedure described in Section 2.4.1, we can convert this policy into a discrete time policy with no higher cost. Note that in the resulting discrete time policy, maintenance actions might not be evenly spaced over the planning horizon but the spacing between subsequent maintenance actions varies by at most one period.

Contrary to the cycle rounding algorithm, the optimal shifted power-of-two policy might perform maintenance on a set of components before maintenance is due on any of the components. While the function $K(\cdot)$ captures the measurable costs associated with a maintenance action, there are inherent risks in performing maintenance on complex systems that are not accounted for in $K(\cdot)$. Thus, performing maintenance before any component has exhausted its cycle limit is likely to be perceived as undesirable in the operational community. Lemma 2.4.8 below shows that, for any feasible schedule, maintenance actions can be deferred until actually needed (i.e., the cycle limit of at least one component has been reached) without increasing the overall cost. Based on that result, the shifted power-of-two schedule can be converted into one in which maintenance is performed only when at least one component has reached its cycle limit.

Lemma 2.4.8 *For any nondecreasing submodular cost function, any feasible schedule can be efficiently converted, without increasing the cost, into one in which no maintenance actions are performed unless some component has zero remaining cycles.*

Proof: Let s be the first period in which we perform maintenance and no component has reached its cycle limit. Let i be the component with the minimum number of cycles remaining at period s and let $\widetilde{f}_i$ be the number of cycles remaining for component i at period s. Note that component i may or may not have been included in the maintenance action scheduled at period s. Consider a new solution in which all maintenance actions in $[s, s + \widetilde{f}_i)$ are shifted to period $s + \widetilde{f}_i$. By the minimality of $\widetilde{f}_i$, it follows that this new solution remains feasible over $[s, s + \widetilde{f}_i]$. All other maintenance actions after period $s + \widetilde{f}_i$ remain unchanged, and, as we have potentially shifted maintenance actions in the interval $[s, s + \widetilde{f}_i]$ later in time, the schedule over $[s + \widetilde{f}_i, \infty)$ remains feasible. Now observe that the submodular cost structure implies the potential consolidation of maintenance actions at period $s + \widetilde{f}_i$ cannot increase the overall cost. ■

The cyclic property of the shifted power-of-two algorithm is lost by delaying maintenance actions until at least one component is in need of repair. However, the resulting schedule remains intuitive as maintenance actions are simply shifted later in time.

As we noted previously, in attaining an improved approximation guarantee, the best shifted power-of-two policy is dependent upon the cost parameters and the component cycle limits. While the cost parameters do not affect the set of shifted power-of-two policies considered, a change in the cost parameters might change the optimal choice of β_i and the resulting shifted power-of-two maintenance schedule. Also, if the cycle limit of a component changes, the β_i coefficient associated with that component might change and so might the best choice of the rounding parameter δ^* from the set of β_i's. Accordingly, there is no a priori bound on the allowable change in the cycle limits as there is for the cycle rounding algorithm. Rather, each possible cycle limit under consideration during the design phase must be evaluated individually.

Shifted Power-of-Two – Bad Example

As was the case for the cycle rounding algorithm, the tight instance for the shifted power-of-two algorithm is pathological and unlikely to arise in practice. It is, however, of interest since it shows that the prior distribution chosen for δ is the best possible distribution and that the analysis cannot be improved by choosing a different distribution for δ. Our example consists of a dependency tree with one module and 2^κ components, where κ is an integer parameter. The maintenance cost for the module is ε and is 1 for each of the components. The cycle limits for the components are 2^κ, $2^\kappa + 1, \ldots, 2^{\kappa+1} - 1$. Figure 2-5 illustrates this example.

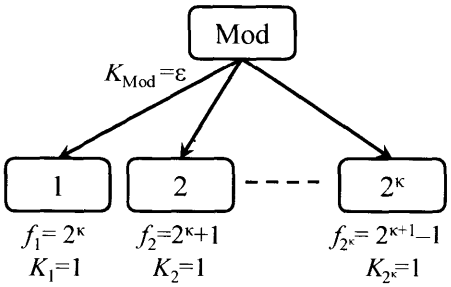

Figure 2-5: Shifted power-of-two bad example.

For ε sufficiently small, the optimal solution will maintain each component exactly at its cycle limit which implies

$$\lim_{\substack{\varepsilon\to 0\\ \kappa\to\infty}} \mathrm{OPT}_{\kappa,\varepsilon} = \lim_{\kappa\to\infty}\left(\frac{1}{2^\kappa} + \frac{1}{2^\kappa+1} + \cdots + \frac{1}{2^{\kappa+1}-1}\right) = \ln(2).$$

From Lemma 2.4.6 we know that the optimal shifted power-of-two rounding parameter δ^* will be of the form $(2^\kappa + d)/2^\kappa$ for an integer d between 0 and $2^\kappa - 1$. For $d = 0$, $f_i^\delta = f_1$ for all components i which implies the long run average cost of the solution will be

$$\frac{K^1}{f_1} + \sum_{i=2}^{2^\kappa} \frac{K^i}{f_1} = \frac{1+\varepsilon}{2^\kappa} + (2^\kappa - 1)\cdot\frac{1}{2^\kappa}.$$

As $\kappa \to \infty$, the first term will converge to 0 while the second term will converge to 1. Therefore, the long run average cost of this solution, if provided by the shifted

power-of-two algorithm, will be 1. For any positive choice of d, the long run average cost of the solution will be

$$\frac{K^1}{f_1^\delta} + \sum_{i=2}^{d} \frac{K^i}{f_i^\delta} + \sum_{i=d+1}^{2^\kappa} \frac{K^i}{f_i^\delta} = \frac{2 + 2 \cdot \varepsilon}{2^\kappa + d} + (d-1) \cdot \frac{2}{2^\kappa + d} + (2^\kappa - d) \cdot \frac{1}{2^\kappa + d}.$$

As $\kappa \to \infty$, the first two terms will converge to 0 for any finite d while the third term will converge to 1. This implies that any solution provided by the shifted power-of-two algorithm will have a long run average cost of 1 while the optimal solution will have a long run average cost of $\ln(2)$.

2.5 Computational Results

While the worst-case analyses for the cycle rounding and shifted power-of-two algorithms ensure that the solutions they provide will be no worse than 2 and $1/\ln(2)$ times the value of the optimal solution, the problem instances for their worst-case examples are significantly different than the instances we would expect to find in practice. Accordingly, we examine the typical empirical performance of these algorithms by testing them on a set of mid-sized problem instances. These instances are large enough to stress the algorithms but small enough to allow for qualitative solution comparison. The approximation algorithms were implemented in MATLAB R2010b run on a standard laptop computer with a 1.9 GHz dual-core processor and 2 GB of RAM.

We conducted extensive computational tests of 15,000 infinite horizon problem instances over a variety of tree structures, cycle limits, and maintenance costs for both the additive and downtime cost functions. These tests were broken down into three groups based on the maintenance costs. In the first group the costs were selected uniformly over a set of values; in the second group, the costs increased linearly as the component cycle limit increased; and for the third group, the costs increased exponentially as the component cycle limit increased. Over this large number of tests, the cycle rounding algorithm had an average runtime of approximately $4 \cdot 10^{-3}$

seconds and yielded solutions no more than 30% above the lower bound for 92% of instances and within 10% percent of the lower bound for 35% of instances. The shifted power-of-two algorithm had an average running time of approximately $4 \cdot 10^{-4}$ seconds with 91% of solutions no more than 30% above the lower bound. For each algorithm and test group, Table 2.5 shows the ratio between the long run average cost and its lower bound as well as the long run average downtime and its lower bound.

Objective	Solution Technique	Uniform Costs	Cost Increasing in Cycle Limit	Costs Exponential in Cycle Limit
Additive	Cycle Rounding	1.13	1.11	1.09
	Shifted Power-of-two	1.24	1.16	1.02
Downtime	Cycle Rounding	1.06	1.25	1.4
	Shifted Power-of-two	1.06	1.23	1.11

Table 2.1: Average solution ratios for 15,000 randomly chosen infinite horizon instances.

Under the additive cost function, the cycle rounding algorithm dominated in 98% of the uniform cost instances while the shifted power-of-two algorithm dominated in 80% of the exponential cost instances. For the instances in which the costs were proportional to the cycle limit, the cycle rounding algorithm produced solutions that had a lower long run average cost in 80% of the instances. Uniform and proportional cost instances yielded no more than 25% variation between the cycle rounding and shifted power-of-two solutions. However, the results were much more drastic in the exponential cost case. In one instance, the shifted power-of-two algorithm had a solution within 1% of the lower bound while the cycle rounding solution was 98% higher than the lower bound.

While there were major differences between the algorithms in test cases with the additive cost function, for test cases with the downtime objective, the outcomes produced by the algorithms were more similar. In fact, in 40% of the uniform cost instances, the cycle rounding solution and the shifted power-of-two solution had the same cost. In only 10% of uniform and proportional cost instances did the cycle rounding and shifted power-of-two solutions vary by more than 15%. Even in the

exponential cost instances in which the shifted power-of-two algorithm outperformed the cycle rounding algorithm in 99% of the instances, there were no extreme cases similar to the instance discussed above.

2.6 Summary

In this chapter, we studied several general models focused on the maintenance of modular systems with component-specific cycle limits. We proposed two easily implemented and conceptually simple frequency-based policies. These policies have provable worst-case guarantees and they perform very close to optimal in computational experiments on a large set of relevant instances. As a by-product of the worst-case analysis, we show that the class of frequency based policies is near optimal. This is particularly important as these policies are operationally intuitive and usually easier to implement in practice.

The algorithms developed in this chapter assume time invariant costs. One natural extension would be to allow the costs to vary over time. With this extension, however, the modular maintenance scheduling problem does not admit a constant factor approximation, even for a tree with one shared module. In Appendix C we show a reduction from set cover to the time varying cost model. Set cover is well known to have an approximation factor no better than $O(\ln(n))$, unless P=NP.

Chapter 3

Graph Visiting and Modular Maintenance

In the last chapter, we developed modeling and algorithmic approaches for maintenance scheduling of modular systems. In this chapter, we consider a version of the same problem which we now cast in the form of a generic combinatorial optimization problem. We when model and analyze the problem as integer and linear programs. The integer programs we consider are used for finite horizon versions of the problem while the linear program we develop allows us to analyze an algorithm for an infinite horizon version of the problem.

Specifically, we examine optimization models and issues associated with a so-called graph visiting problem in which we need to visit each node of a given directed acyclic graph at least as often as a specified frequency. We provide a general mixed integer programming formulation, as well as a formulation that models situations when the solution must be cyclic. We show that when the frequencies are powers of any fixed integer, the linear programming relaxation of the general and cyclic formulations both have integer optimal solutions. Using this fact, we show that the integrality gap of these formulations is at most 2 and that the optimal values for the general and cyclic mixed integer programs differ by a factor of at most 2. Finally, using a linear programming based analysis, we show that a so-called a-priori shifted power-of-two algorithm has a worst case performance of $1/\ln(2) \approx 1.4427$. The optimal

solution to the linear program shows that the worst case guarantees is tight. This analysis matches with the analytical work in Chapter 2 for the closely related shifted power-of-two algorithm.

3.1 Introduction

Building upon the work in Chapter 2, we introduce a combinatorial optimization problem we call "The Graph Visiting Problem." This problem has its roots in maintenance scheduling for modular systems comprised of components that must be repaired or replaced at least as often as a component specific frequency. The objective is to minimize the total or long run average cost for finite and infinite planning horizons, respectively.

Let $G = (\mathcal{N}, E)$ be a rooted and directed acyclic graph (dependency graph) with nodes $\mathcal{N}$ and a designated root node, which is the only node without incoming arcs. We assume that G has one or more directed paths from the root node to every other node and that each node i has an associated *cycle limit* (or frequency) f_i, and associated cost K_i. Over a discrete planning horizon of T periods, we need to "visit" each node i (to repair it) at least as often as its associated frequency. That is, if we visit node i in period t we must visit it again on or before period $t + f_i$. When we visit node i, we must also visit each node on every path from the root node to that node, which we refer to as its ancestors $\mathcal{A}_i \subseteq \mathcal{N}$. We assume that the set $\mathcal{A}_i$ includes node i as well as the root. We also refer to any node as a successor of its ancestors. We seek a visiting schedule that minimizes total costs, which are developed in detail below.

For any set S of nodes, let $K(S)$ denote the sum of the costs for all the nodes in S. That is $K(S) = \sum_{i \in S} K_i$. We can cast the underlying combinatorial optimization problem as follow:

The Graph Visiting Problem

Given a dependency graph G, decide for each period t on a set S_t of nodes to be visited so that the total cost over the planning horizon, $\sum_{t=1}^{T} K(S_t)$, is minimized.

The node sets $S_1, S_2, \ldots, S_T$ are required to satisfy the following feasibility conditions:

(i) node i is included in at least one of the sets $S_1, S_2, \ldots, S_{f_i}$,

(ii) if node i belongs to S_t, then so does all its ancestors $\mathcal{A}_i$, and

(iii) if, for some period t, node i belongs to S_t, then node i also belongs to S_τ for some $\tau \leq t + f_i$.

As a notational convenience, and in keeping with the prior motivation, whenever $i \in S_t$ we say that we visit node i at time period t.

Note that by property (ii), in each time period t, S_t induces a rooted subgraph of G. The complexity of this problem arises due to the possibility of cost sharing between nodes, as their ancestor node sets intersect, which motivates the combination of visitation actions into common time periods. Intuitively, if there is a meaningful overlap between the ancestors of i_1 and i_2, we would prefer to jointly schedule these nodes. On the other hand, scheduling node visits too frequently will drive up the overall number of visits along the entire planning horizon, and consequently, the resulting cost.

In this chapter, we examine optimization related models and issues associated with the graph visiting problem. In Section 3.2 we review literature related to the graph visiting problem. We then develop preliminary results in Section 3.3. We provide a general mixed integer programming (MIP) formulation, Section 3.4, as well as a formulation that models situations when the solution must be *cyclic* (i.e., nodes are visited at a fixed frequency over the planning horizon), Section 3.5. Using these formulations, in Section 3.6 we show that:

- For a special case in which the frequencies are *nested*, and in particular powers of 2, the linear programming relaxation of the general and cyclic formulations both have integer optimal solutions.
- The integrality gap of each formulation is at most 2.

- The optimal values of the general and cyclic MIPs differ by a factor of at most 2.

Finally in Section 3.7, we examine a linear programming formulation that provides the worst case performance guarantee of a cyclic rounding heuristic, the so-called a-priori shifted power-of-two algorithm. This a-priori algorithm computes a predetermined number of cyclic visitation schedules that do not depend upon the given instance. As the number of predetermined visitation schedules increases, the minimum cost schedule for a given instance is shown to have a cost at most $1/\ln(2)$ times the cost of the optimal visitation schedule. We also provide worst case instances that attain the worst-case guarantee for a chosen set of visitation schedules. This analysis matches the analytical work of the previous chapter that obtains a worst-case bound of $1/\ln(2)$ for the closely related shifted power-of-two algorithm.

The previous chapter was motivated by applications in maintenance scheduling of modular systems. In this context, the dependency graph G is assumed to be a directed tree whose leaf nodes are components of a modular system that must be accessed and repaired. The interior nodes are modules that house the components (or other modules). The tree structure arises since the removal of one module requires the removal of other modules (its ancestors). For instance, the F100 engine, a US Air Force engine used in fighter aircraft, is composed of five major modules: the fan module, the core module, the fan drive turbine module, the augmentor/exhaust module, and the gearbox module. If maintenance is required on a component in the fan drive turbine module, the augmentor/exhaust module must be removed to access the fan drive turbine module Forbes and Wyatt [21].

Scheduled maintenance activities for modular systems, such as an aircraft engine, are driven by usage limits (cycle limits) that specify the maximum number of periods of use between subsequent maintenance actions. For example, a cycle for the starter system in an aircraft engine could be one startup sequence. For components in an aircraft braking system, a cycle could be one landing sequence. Each component can be used for a certain number of cycles and then must be repaired or replaced due to safety or failure concerns. These cycle limits are determined through a number of

methods including physical testing, simulation, and analytical assessment. Although it is possible that components fail prior to their cycle limits, due to the conservative nature of these cycle limits, such events are extremely rare. As a result, it is common to assume that a component is operational until its cycle limit is reached and that after maintenance it again has a full cycle limit. It is these cycle limits that will determine the required visitation frequency on the graph.

There are several practical settings within the U.S. Air Force in which the models and algorithms studied in this chapter, that work for directed acyclic dependency graphs, could be of use. During the design and testing phase of a modular system, a development team could use the models and algorithms we discuss to explore the short and long term implications of changing various design parameters such as early investment in maintenance capability to reduce per incident maintenance costs or increasing component cycle limits to decrease the future sustainment costs for the system. Once a system has been fielded, operational considerations or challenges might affect the maintenance philosophy codified during the development phase. During this operational phase, actual maintenance costs or costs from an auxiliary optimization problem (i.e., linear programming dual variables) could be used to produce feasible and low cost maintenance schedules. The models and algorithms studied in this chapter could also be used as building blocks within more comprehensive operational maintenance and inventory management models.

3.2 Related Literature

Although, to our knowledge, this is the first work on the graph visiting problem, the problem we consider is rooted in a modular maintenance scheduling problem discussed in Chapter 2. As mentioned, there has been relatively little work addressing frequency constrained maintenance scheduling with multiple fixed cost setup activities, especially for finite horizon problems. van Dijkhuizen [47] and van Dijkhuizen and van Harten [48] both consider the special case in which G is a rooted tree and restrict attention to cyclic policies in which maintenance occurs at integer multiples

of a selected, node-dependent, maintenance frequency. van Dijkhuizen [47] models and solves the finite horizon problem via an integer programming formulation. van Dijkhuizen and van Harten [48] addresses the infinite horizon problem via a branch and bound algorithm. Neither work addresses the cost implications associated with limiting attention to cyclic schedules and neither are polynomial time solution methods. In addition, their work is limited to the additive cost function induced by the tree structure. The MIP formulations presented in Sections 3.4 and 3.5 can be used for any rooted, directed, acyclic graph. In addition, the a-priori shifted power-of-two results presented in Section 3.7 apply to a more general class of submodular cost functions.

The main contribution of the previous chapter was the development of two infinite horizon approximation algorithms, cycle rounding and shifted power-of-two, with worst case guarantees of 2 and $1/\ln(2)$, respectively. In addition, we showed computationally that cyclic schedules are near optimal for infinite horizon instances. Using the general and cyclic MIP formulations, we will show that cyclic schedules are also near optimal for finite horizon problems. These analysis also show that the cycle rounding algorithm proposed in the previous chapter is a 2-approximation algorithm for finite horizon instances. Finally, we extend the shifted power-of-two algorithm to instances in which the cycle limits are initially unknown when the shifting parameters are chosen but are revealed after the shifted parameters have been chosen. By choosing a sufficiently large number of shifting parameters, we obtain the same approximation guarantee as the previous analysis that used the cycle limits to find the optimal shifting parameter. While we used a randomized rounding technique to prove the approximation results for the shifted power-of-two algorithm, we use will use linear programming duality to obtain the same result for the a-priori shifted power-of-two algorithm.

3.3 Preliminaries

Consider any feasible solution to the graph visiting problem. Due to the frequency requirement, each node i must belong to at least $\lfloor T/f_i \rfloor$ of the sets $S_1, S_2, \ldots, S_T$. Accordingly, the cost of any feasible solution is bounded from below as:

$$\sum_{t=1}^{T} K(S_t) = \sum_{t=1}^{T} \sum_{i \in S_t} K_i \geq \sum_{i \in \mathcal{N}} \left\lfloor \frac{T}{f_i} \right\rfloor \cdot K_i. \tag{LB}$$

This simple lower bound will be useful at several points in the analysis.

Since, whenever we visit node i we must visit all its ancestors $\mathcal{A}_i$, without loss of generality we can assume that if $j \neq i$ belongs to $\mathcal{A}_i$, then $f_j \leq f_i$. That is, without changing any feasible solution to the problem, or its cost, we can set f_j as the minimum of itself and the frequencies of all the nodes that are its successors in G (i.e., the new value of f_j is set to $\min_i\{f_i : j \in \mathcal{A}_i\}$). An important consequence of this observation is that the node frequencies along any path from a node to the root are non-increasing.

For convenience, let us order the nodes as $1, 2, \ldots, |\mathcal{N}|$ in non-decreasing order of frequencies so that $f_i \leq f_j$ whenever $i < j$. Without loss of generality we can assume that $1 < f_i \leq T$ for all nodes i. If $f_i = 1$ for some node i, then node i, and all of its ancestors $\mathcal{A}_i$, can be included in every visitation set over the planning horizon and removed from the problem. If $f_i > T$ for some node i, then node i need not be visited during the planning horizon and it and all of its successors, can be discarded from the problem.

3.3.1 Nested Frequencies

Suppose that the cycle limits are *nested* in the sense that for each i, f_i/f_{i-1} is an integer. Consider a schedule in which we visit each node i at all times that are a multiple of it frequency, that is, at times $f_i, 2 \cdot f_i, \ldots, \lfloor T/f_i \rfloor \cdot f_i$. Let $S_1^*, S_2^*, \ldots, S_T^*$ be the resulting solution. This solution is feasible since whenever $i \in S_t^*$, all the nodes with lower or equal frequencies, and in particular all its ancestors in $\mathcal{A}_i$, will

belong to S_t^*. This is a direct result of the modification to the frequencies above and in particular the fact that the frequencies are non-increasing along any path from a node to the root.

Note that the solution $S_1^*, S_2^*, \ldots, S_T^*$ visits each node the minimum number of times possible over the planning horizon which implies that its cost equals the lower bound (LB) and thus is optimal. This observation yields the following result.

Theorem 3.3.1 *If the node frequencies are nested, then it is optimal to visit each node i at all times in $[1, T]$ that are multiple of its frequency f_i.*

It is also important to note that if the frequencies are all powers of any given positive integer b, or are all of the form $a \cdot b^q$ for some given positive integers a and b and varying positive integer q, then they will be nested.

3.3.2 Power of 2 Based Heuristics

Let a and b be given positive integers and let P be any instance of the graph visiting problem. Suppose we create a new "rounded" instance by replacing the frequency of each node in P by rounding it down the nearest value of $a \cdot b^q$ for some value of q. That is, for a node i with $a \cdot b^q \leq f_i < a \cdot b^{q+1}$, we set its rounded frequency to $a \cdot b^q$. By Theorem 3.3.1, and the fact that the rounded problem is nested, it is optimal to visit each node i at all times that are multiples of its rounded frequency $a \cdot b^q$. This approach provides an approximate solution to instance P. For a given value of a, we refer to the rounded down solution as a shifted power-of-b heuristic. Clearly, there are infinitely-many shifted power-of-b heuristics, depending on the value of a.

Let us make one other observation. Suppose P1 and P2 are two instances of the graph visiting problem defined on the same rooted graph G, and further that the frequency of each node in P1 is less than or equal to the frequency of that node in P2. Then, since every feasible solution to P1 is feasible to P2, the optimal cost of P2 is less than or equal to the optimal cost of P1. Combined with Theorem 3.3.1, this observation implies the following result.

Lemma 3.3.2 *Let a and b be positive integers and suppose that each node frequency f_i in the graph visiting problem is at least $a \cdot b$. Then the shifted power-of-b heuristic is a b-approximation algorithm.*

Proof : Let P1 be the graph visiting problem on the same graph as P but with rounded down frequencies according to the power-of-b rounding. Let f_i^r denote the (rounded down) frequency of node i in P1. From Theorem 3.3.1, in the rounded problem it is optimal to visit each node i at all multiples of f_i^r. By definition, for some integer $q \geq 1$, $f_i^r = a \cdot b^q \leq f_i < a \cdot b^{q+1}$, implying that $f_i / f_i^r < b$.

Therefore, for every visit to node i in any feasible solution to the graph visiting problem P, the optimal solution to the rounded problem P1 visits node i at most $b-1$ times. After the final visit of node i in any feasible solution to problem P, the optimal solution to the rounded problem P1 might visit node i at most $b-1$ additional times. Therefore, since $\lfloor T/f_i \rfloor \geq 1$ by assumption,

$$\left\lfloor \frac{T}{f_i^r} \right\rfloor \leq (b-1) \cdot \left\lfloor \frac{T}{f_i} \right\rfloor + (b-1) \leq b \cdot \left\lfloor \frac{T}{f_i} \right\rfloor .$$

Consequently, the cost of the shifted power-of-b solution to P1, OPT (P1), and the cost of the optimal solution to the original problem P, OPT (P), are related as follows:

$$\text{OPT (P1)} = \sum_{i \in \mathcal{N}} K_i \cdot \left\lfloor \frac{T}{f_i^r} \right\rfloor \leq b \cdot \sum_{i \in \mathcal{N}} K_i \cdot \left\lfloor \frac{T}{f_i} \right\rfloor \leq b \cdot \text{OPT (P)},$$

The last inequality follows from the lower bound (LB). ∎

We will focus specific attention in the first part of this chapter on the special case when $a = 1$ and $b = 2$, and thus is a 2-approximation algorithm. We will refer to this as the power-of-two algorithm. In the second portion of the chapter, we will again consider an algorithm in which $b = 2$. However, we will allow a to be a real number in the interval $[1, 2)$ and will refer to this version as a shifted power-of-two algorithm.

Finally, it is of note that using a redefined notion of K_i and LB, the results from Theorem 3.3.1 and Lemma 3.3.2 hold for any submodular cost function. This

connection is explored further in Section 3.7.

3.4 General MIP Formulation

To formulate the graph visiting problem as an integer program, let the variables y_s^i indicate the inclusion of node i in set S_s. To ensure that the optimal schedule visits node i at least once every f_i time periods, we specify that at each time period t, node i must have been visited on or after time period $t - f_i + 1$. Let variable x_{st}^i indicate that in preparation for period t, node i was last visited in period s.

$$\text{minimize} \sum_{i \in \mathcal{N}} \sum_{s=1}^{T} K_i \cdot y_s^i \tag{P}$$

subject to

$$\sum_{s=t-f_i+1}^{t} x_{st}^i = 1, \qquad \forall\, i \in \mathcal{N}, f_i \leq t \leq T, \tag{3.1}$$

$$x_{st}^i \leq y_s^k, \qquad \forall\, i \in \mathcal{N}, k \in \mathcal{A}_i, 1 \leq s \leq t \leq T, \tag{3.2}$$

$$x_{st}^i \geq 0, \qquad \forall\, i \in \mathcal{N}, 1 \leq s \leq t \leq T, \tag{3.3}$$

$$y_s^i \in \{0,1\}, \qquad \forall\, i \in \mathcal{N}, 1 \leq s \leq T. \tag{3.4}$$

The objective function minimizes the cumulative visitation costs. Constraints (3.1) ensure that each node is visited within the frequency requirement. Constraints (3.2) ensure node i belongs to set S_s only when all ancestor nodes also belong that set. Constraints (3.3) enforce nonnegativity for the visitation variables while Constraints (3.4) force the visitation decision variables to be binary. We do not need to force x_{st}^i to be binary. The model (P) always has an optimal solution in which each x_{st}^i is binary. In particular, set $x_{st}^i = 1$ for the latest period s for which $y_s^i = 1$ and set $x_{s't}^i = 0$ for all $s' \neq s$.

Similar to the combinatorial lower bound (LB), we can lower bound the cost of the optimal solution to (P), specifically the linear programming relaxation of (P).

Lemma 3.4.1 *The optimal solution to the linear programming relaxation of* (P)

(OPT L(P)) *has a cost greater than or equal to:*

$$\text{OPT L(P)} \geq \sum_{s=1}^{T} K_i \cdot \left\lfloor \frac{T}{f_i} \right\rfloor .$$

Proof : Let (x^*, y^*) be an optimal solution to L(P), the linear programming relaxation of (P). For every node i, divide the planning horizon into $\lfloor T/f_i \rfloor$ intervals $(I_1, I_2, \ldots I_\Psi)$, with the first $\lfloor T/f_i \rfloor - 1$ of length f_i. The last interval, I_ψ will have at least f_i periods. The feasibility of any fractional solution implies that for every $1 \leq \psi \leq \Psi$ and t being the last time period in I_ψ,

$$\sum_{s \in I_\psi} x_{st}^{*i} \geq 1.$$

Constraint (3.2) of the model (P) then implies that

$$\sum_{s \in I_\psi} y_s^{*i} \geq \sum_{s \in I_\psi} x_{st}^{*i}, \quad \forall\, i \in \mathcal{N}.$$

Using the fact that there are $\lfloor T/f_i \rfloor$ intervals that partition the planning horizon,

$$\sum_{1 \leq s \leq T} y_s^{*i} = \sum_{\psi \in 1..\Psi} \sum_{s \in I_\psi} y_s^{*i} \geq \left\lfloor \frac{T}{f_i} \right\rfloor .$$

Multiplying these lower bounds on y_s^{*i} by the corresponding costs K_i yields the result. ■

3.4.1 MIP Challenges

Using the power-of-two heuristic, we will later bound the integrality gap of this formulation by a factor of 2. Even though the formulation has a bounded integrality gap, it is similar in structure to the joint replenishment problem MIP [28] and it is unlikely that a mixed-integer programming solver will be able to solve the problem efficiently for realistic sized instances. In computational experiments, a commercial

mixed-integer programming solver was not able to solve instances with 50 nodes and a planning horizon of 30 periods within 24 hours. If we are interested in approximating the infinite horizon problem, this formulation would become even more intractable as T increases.

In addition to prohibitively long solution times, an optimal solution to the graph visiting problem can be very complex, unintuitive, and highly dependent upon node visitation costs and the planning horizon length. For example, an optimal solution of a small instance with 12 nodes and a planning horizon of 30 periods, visited a node with a cycle limit of 3 at the odd periods from 3 to 15 and then visited the node at the even periods from 18 to 30. The transition from 15 to 18 results from the interaction of this node with other nodes that have longer cycle limits and the cost structure that ties them together. In addition, relatively small changes to the node visitation costs and the length of the planning horizon for this small instance resulted in significant changes to the visitation schedule. In a number of small instances tested, the optimal visitation schedules were not stationary, nor cyclic, which results in complex operational issues for implementation. To develop operationally tenable schedules that lend themselves better to implementation and operational codification, we next consider optimal cyclic policies.

3.5 Optimal Cyclic Policies

The formulation used by van Dijkhuizen [47] yields cyclic schedules and based on his computational testing, it might be tractable even for very large instances of the graph visiting problem. Our model differs from van Dijkhuizen's original formulation because it does not have objective function costs associated with assigning a node to

a given frequency.

$$\text{minimize} \sum_{i \in \mathcal{N}} \sum_{s=1}^{T} K_i \cdot y_s^i \tag{CP}$$

subject to

$$\sum_{\phi=1}^{f_i} x_\phi^i = 1, \qquad \forall\, i \in \mathcal{N}, \tag{3.5}$$

$$x_\phi^i \leq y_s^k, \qquad \forall\, i \in \mathcal{N}, k \in \mathcal{A}_i, 1 \leq \phi \leq f_i, \; 1 \leq s \leq T : s \bmod \phi = 0, \tag{3.6}$$

$$x_\phi^i \geq 0, \qquad \forall\, i \in \mathcal{N}, 1 \leq \phi \leq f_i, \tag{3.7}$$

$$y_s^i \in \{0, 1\}, \qquad \forall\, i \in \mathcal{N}, 1 \leq s \leq T. \tag{3.8}$$

As in the general MIP formulation (P), the variables y_s^i indicate the inclusion of node i in the set S_s. Variables x_ϕ^i denote the assignment of node i to a visitation frequency of once every ϕ periods. Note again that, as mentioned in Section 3.3, $1 < f_i \leq T$ and thus the cyclic formulation will be smaller than the general formulation. The objective function minimizes the cumulative visitation costs. Constraints (3.5) ensure the visitation frequency assigned to a node is less than or equal to its cycle limit. Constraints (3.6) ensure that the dependencies for a node are satisfied for each time period that is a multiple of the chosen visitation frequency. Constraints (3.7) enforce nonnegativity of the frequency assignment variables, and Constraints (3.8) force the visitation decision variables to be either zero or one. As in the formulation (P), we need only nonnegativity constraints for the x variables. An optimal binary solution for the x variables can easily be determined given a feasible solution to (CP) with binary y variables and nonnegative x variables by setting $x_\phi^i = 1$ for the largest value of ϕ for which $x_\phi^i > 0$ and setting all others to zero for that node.

Note, however, that (CP) can be strengthened by replacing constraints (3.6) with

$$\sum_{\substack{\phi=1: \\ s \bmod \phi = 0}}^{f_i} x_\phi^i \leq y_s^k, \qquad \forall\, i \in \mathcal{N}, k \in \mathcal{A}_i, 1 \leq s \leq T. \tag{3.9}$$

This change in the formulation significantly improves the linear programming relaxation of (CP) and results in integer optimal solutions in many instances, as indicated below. The improved formulation is not, however, integral.

While the solutions provided by (CP) are operationally attractive, the cost consequences associated with implementing cyclic policies rather than general policies need to be determined. As a consequence of the integrality gap analysis for (P), we will show that the cost of the optimal solution to (CP) is no more than twice the cost of the optimal solution to (P). Computational tests indicate that the cost consequences we can expect in practice are much better than this worst case guarantee. In particular, in 200 instances tested, the largest observed difference between the optimal linear programming relaxation of (P) and the optimal integral solution to (CP) was 10%, with ninety percent of the differences less than 7% and an average difference of 4%. Due to limited computation time, we did not calculate the optimal integral solution for (P). Since the linear programming relaxation of (P) is a lower bound on the cost of the optimal integral solution to (P), the differences between the cyclic and non-cyclic solutions can be only better than those indicated.

It is easy to see that every feasible solution to the linear programming relaxation of (CP) corresponds to a feasible solution of equal cost to the linear programming relaxation of (P). To make the association, given a solution $(\overline{y}, \overline{x})$ to the linear programming relaxation of (CP), construct a solution (y, x) to the linear programming relaxation of (P) as follows:

$$y_s^i = \overline{y}_s^i, \qquad \forall\, i \in \mathcal{N}, 1 \le s \le T, \tag{3.10a}$$

$$x_{st}^i = \begin{cases} \displaystyle\sum_{\substack{\phi=1: \\ s \bmod \phi = 0}}^{f_i} \overline{x}_\phi^i, & \forall\, 1 \le s \le t \le s+\phi \le T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.10b}$$

Let OPT (P) be the cost of the optimal integral solution to (P) and OPT L(P) be the cost of the optimal solution to the linear programming relaxation of (P). OPT (CP) and OPT L(CP) are defined similarly. As a consequence of the prior

construction, OPT L(P) $\leq$ OPT L(CP). Also, when $\overline{y}$ is integer so is y, and therefore, OPT (P) $\leq$ OPT (CP).

In contrast to the long solution times for the original MIP formulation (P), the cyclic MIP (CP) solves quickly for both the original and modified formulations with constraints (3.9) in place of (3.6). While both solve quickly, the tightness of the formulations differs substantially. In none of the 200 instances tested did the linear programming relaxation for the original cyclic formulation, with constraints (3.6), result in an integer solution. The average integrality gap for these instances was 10%. In 192 of the same instances, the improved formulation using constraints (3.9) had integer optimal solutions. The average integrality gap of the 8 non-integral instances was 0.08%.

3.6 Integrality Gaps of (P) and (CP)

We begin by considering the power-of-two heuristic discussed in Section 3.3. We will show that it provides an optimal solution to the linear programming relaxations of (P) and (CP) when the cycle limits are all powers of 2. Using that fact, we can then bound the integrality gap of the original problem. As previously noted, we assume without loss of generality that the nodes are numbered in nondecreasing order of their cycle limits, that is $f_i \leq f_j$ for every $i < j$.

Theorem 3.6.1 *If the cycle limits are nested, then the linear programming relaxations of* (P) *and* (CP) *both have optimal integral solutions.*

Proof : Let $S_1, S_2, \ldots, S_T$ be a cyclic solution in which we visit each node i at all times that are multiple of its cycle limit, that is, at times $f_i, 2 \cdot f_i, \ldots, \lfloor T/f_i \rfloor \cdot f_i$. Let $\overline{y}_s^j = 1$ if $i \in S_s$. Then $\overline{y}$ corresponds to a visitation frequency assignment $\overline{x}$ and is a feasible solution to (CP). As noted before, the cost of this solution is given by

$$\sum_{t=1}^{T} K(S_t) = \sum_{t=1}^{T} \sum_{i \in S_t} K_i = \sum_{i \in \mathcal{N}} \left\lfloor \frac{T}{f_i} \right\rfloor \cdot K_i$$

The final equality follows from the construction of the policies.

As a result of Lemma 3.4.1, any solution to the linear programming relaxation costs at least as much as the cost of the nested solution $(\overline{y}, \overline{x})$ which is integral. Therefore, the associated integral solution (y, x) from (3.10) is optimal for the linear programming relaxation L(P). Finally, L(P) $\leq$ L(CP) implies:

$$\sum_{t=1}^{T} K(S_t) = \text{OPT L(P)} \leq \text{OPT L(CP)} \leq \text{OPT (CP)} \leq \sum_{t=1}^{T} K(S_t).$$

Since all of the inequalities must hold with equality, $(\overline{y}, \overline{x})$ is an integral optimal solution of the linear programming relaxation of (CP) and the associated (y, x) is an integral optimal solution of the linear programming relaxation of (P). ■

Using this fact as applied to the power-of-two rounding, we can now bound the integrality gap for the two formulations.

Theorem 3.6.2 *The integrality gaps of* (P) *and* (CP) *are at most 2.*

Proof : Let (x^*, y^*) be an optimal solution to L(P), the linear programming relaxation of (P) and let (x, y) be an optimal integral solution to (P). Let f_i^r be the rounded down power of 2 cycle limit for node i and $(\widehat{x}, \widehat{y})$ be the resulting solution to the rounded down problem. In the rounded solution, the contribution of each node i to the objective function is

$$K_i \cdot \sum_{s=1}^{T} \widehat{y}_s^i.$$

Divide the planning horizon T into intervals of length f_i, with the final interval strictly less than f_i. The contribution to the objective function by each component i in the rounded solution can be bounded as follows:

$$\begin{aligned} K_i \cdot \sum_{s=1}^{T} y_s^{*i} \leq K_i \cdot \sum_{s=1}^{T} y_s^i \leq K_i \cdot \sum_{s=1}^{T} \widehat{y}_s^i \\ \leq K_i \cdot \left(\left(2 \cdot \left\lfloor \frac{T}{f_i} \right\rfloor - 1 \right) + 1 \right) \leq K_i \cdot 2 \cdot \sum_{s=1}^{T} y_s^{*i}. \end{aligned}$$

The first inequality follows from the definitions of y^* and y. The second inequality follows from the fact that $\widehat{y}$ defines a feasible integral solution to (P) as shown above. The subsequent inequality follows from the fact that during the first $\lfloor T/f_i \rfloor$ intervals, the power-of-two rounding will visit node i at most $(2 \cdot \lfloor T/f_i \rfloor - 1)$ times. The remaining interval has a length strictly less than $2 \cdot f_i^r$. Thus, the power-of-two rounded solution will visit node i at most once in that interval. The final inequality follows from the same argument as (LB).

Summing the prior bound over all nodes i shows that

$$\text{OPT L(P)} \leq \text{OPT (P)} \leq \sum_{i \in \mathcal{N}} K_i \cdot \sum_{s=1}^{T} \widehat{y}_s^i \leq 2 \cdot \sum_{i \in \mathcal{N}} K_i \cdot \sum_{s=1}^{T} y_s^{*i} = 2 \cdot \text{OPT L(P)}.$$

Since any feasible solution to the linear programming relaxation of (CP) can be translated into a feasible solution, of equal cost, to the linear programming relaxation of (P) we know that:

$$\begin{aligned} \text{OPT L(P)} \leq \text{OPT L(CP)} \leq \text{OPT (CP)} &\leq \sum_{i \in \mathcal{N}} K_i \cdot \sum_{s=1}^{T} \widehat{y}_s^i \\ &\leq 2 \cdot \sum_{i \in \mathcal{N}} K_i \cdot \sum_{s=1}^{T} y_s^{*i} = 2 \cdot \text{OPT L(P)}. \end{aligned}$$

■

In addition, the power-of-two rounding also allows us to bound the cost increase incurred by restricting attention to cyclic solutions.

Corollary 3.6.3 *The optimal cyclic solution has a cost at most twice that of the optimal, potentially non-cyclic, solution:* OPT (P) $\leq$ OPT (CP) $\leq 2 \cdot$ OPT (P).

This is of particular interest as cyclic solutions are much easier to implement in practice.

In addition to the additive costs presented above, it is interesting to consider the case when the cost incurred in any time period is the maximum cost of any node in the visitation set. In particular, $K(S_t) = \max_{i \in S_t} K_i$. Based on the maintenance

scheduling roots of the problem, we refer to this as the downtime cost function. While the formulations (P) and (CP) will change, similar analysis for the downtime cost function yields the following results.

Observation 3.6.4 *If the cycle frequencies are nested, then the linear programming relaxations of the corresponding general and cyclic formulations both have optimal integral solutions.*

Observation 3.6.5 *The integrality gaps of the corresponding general and cyclic formulations are at most 2.*

Observation 3.6.6 *The optimal cyclic solution has a cost at most twice that of the optimal, potentially non-cyclic, solution.*

3.7 A-priori Shifted Power-of-Two

While the prior formulations could be used for finite horizon problems, solving the mixed integer programs for large values of T to approximate an infinite horizon instance is computationally intractable. In addition, these formulations assume additive costs. As in the previous chapter, in this section we consider an infinite horizon model with more general submodular costs $K(S)$ and a rounding scheme in which the cycle limits are rounded down to a specified rounding parameter times the next lower power of 2. In their work, they decompose the node cycle limits into a product of the next lower power of 2 and a coefficient β_i between 1 and 2, $f_i = \beta_i \cdot 2^{\kappa_i}$. They show that the optimal rounding parameter must equal one of the β_i values. Accordingly, the optimal rounding can be found by performing the rounding for each of the β_i values, evaluating the costs, and choosing the schedule with the minimum cost. Such an algorithm yields a $1/\ln(2) \approx 1.4427$ approximation for infinite horizon instances. In contrast to that work, we will assume that the cycle limits are unknown when the rounding parameters are chosen and will empirically, and then analytically, establish the same bound. The resulting schedule may be continuous but, by the same argu-

ment at the end of Section 2.4.1, can be converted to a discrete time schedule without increasing the cost.

Definition 3.7.1 (Submodular function) *A real-valued function $K : 2^{\mathcal{C}} \rightarrow \mathbb{R}$, defined on subsets of a ground set $\mathcal{C}$, is submodular if for any two subsets, S and S',*

$$K(S) + K(S') \geq K(S \cup S') + K(S \cap S').$$

An equivalent definition of a submodular function focuses on economies of scale. A real-valued function defined on subsets of a ground set $\mathcal{C}$ is submodular if for every $S \subset S' \subset \mathcal{C}$ and $i \in \mathcal{C} \setminus S'$:

$$K(S \cup \{i\}) - K(S) \geq K(S' \cup \{i\}) - K(S').$$

We assume that $K(S)$ is nondecreasing and that $K(\emptyset) = 0$.

3.7.1 Empirical Performance Guarantee

We begin by empirically establishing the performance guarantee for rounding parameters that evenly divide the interval $[1, 2)$. In particular, if two rounding parameters are used, they will be $\{1, 3/2\}$. For three rounding parameters they will be $\{1, 4/3, 5/3\}$. In addition, we will use a linear programming formulation to find both the approximation ratio and the corresponding worst case instance for any number of discretizations.

Let D be the set of intervals induced by the chosen rounding parameters. From the example above for two rounding parameters, $D = \{[1, 3/2), [3/2, 2)\}$. In addition, let S^d be the set of nodes whose cycle limits fall into the d^{th} interval. If we consider two rounding parameters, $\{1, 3/2\}$, S^1 will contain those nodes with a cycle limit in the interval $2^{\kappa_i} \leq f_i < 3/2 \cdot 2^{\kappa_i}$ and S^2 those with a cycle limit $3/2 \cdot 2^{\kappa_i} \leq f_i < 2^{\kappa_i + 1}$.

Note that the rounded schedules will be cyclic and nested. As established pre-

viously in Lemma 3.3.2, such a scheme using one rounding parameter, namely $\{1\}$, yields an approximation guarantee of 2. In the case of two rounding parameters $\{1, 3/2\}$, if the chosen rounding parameter, δ, is 1, the maximum increase in frequency for the two sets, S^1 and S^2, are 3/2 and 2, respectively. If $\delta = 3/2$, the maximum increase in frequency for the two sets, S^1 and S^2, are 2 and 4/3, respectively. This would seem to imply that the approximation guarantee for two rounding parameters is again 2. However, we will show that the worst case distribution of costs cannot attain this bound.

To extend (LB) to general submodular cost functions, consider the same charging scheme used in the previous chapter in which node i is charged for the marginal cost of adding i to a visitation set that includes only lower indexed nodes. For any set S of nodes, let $S_i = S \cap \{1, 2, \ldots, i-1\}$ be the subset of nodes in S with a cycle limit that is lower than or equal to the cycle limit of node i, which by definition have a lower index. We define $S_i = \{\emptyset\}$ if i is the lowest indexed node in the set S. We then charge node i with the marginal additional cost of adding node i to the set S_i. That is, in the cost $K(S)$ of node set S, node $i \in S$ is charged $K(S_i \cup \{i\}) - K(S_i)$. The cost $K(S)$ is then the sum of the charges of all the nodes that it contains. By the submodularity of $K(\cdot)$, for any node set S and for any i,

$$K(S_i \cup \{i\}) - K(S_i) \geq K(\{1, 2, \ldots, i-1\} \cup \{i\}) - K(\{1, 2, \ldots, i-1\}).$$

We define $K^i = K(\{1, 2, \ldots, i-1\} \cup \{i\}) - K(\{1, 2, \ldots, i-1\})$ which we refer to as the *residual cost* of node i. By the submodularity of $K(\cdot)$, K^i is the minimal charge node i can incur each time it appears in a node set S and thus the lower bound (LB) is valid with K_i replaced with K^i. Note that in the additive case $K_i = K^i$.

At each visitation in any shifted power-of-two schedule, each node contributes only its residual cost, K^i, (i.e., the marginal cost of adding i to a visitation set that includes all lower indexed nodes). We will define R_d as the fraction of residual costs

for nodes with a cycle limit that fall in the d^{th} interval. That is:

$$R_d = \frac{\sum_{i \in S^d} K^i}{\sum_{i \in \mathcal{N}} K^i}$$

To determine the worst case performance guarantee for a given number of intervals, we will determine the worst case distribution of residual costs over the intervals. For the case of two intervals, this calculation becomes

$$\max_{\substack{R_1, R_2 \geq 0: \\ R_1 + R_2 = 1}} \min \left\{ \frac{3}{2} \cdot R_1 + 2 \cdot R_2, 2 \cdot R_1 + \frac{4}{3} \cdot R_2 \right\}$$

The first term inside the minimum is the cost incurred when the rounding parameter δ equals 1 and the second when δ equals 3/2. The maximum occurs when the two terms are equal which yields a system of two equations with two unknowns. By solving this system of equations, we find that the worst case distribution of residual costs is $R_1 = 4/7$ and $R_2 = 3/7$ which implies a performance guarantee of 12/7. The tight instance for this algorithm is two nodes with residual costs of 4 and 3 with cycle limits sufficiently close to $3/2 \cdot 2^\kappa$ and $2^{\kappa+1}$, respectively.

A key insight from this example is that to attain the worst case instance, we need to maximize the minimum of the terms while ensuring that the sum of the residual cost distributions is 1. Accordingly, we can formulate a linear program that solves the problem.

maximize Z (DLP-2)

subject to

$$R_1 + R_2 = 1, \tag{3.11}$$

$$Z \leq \frac{3}{2} \cdot R_1 + 2 \cdot R_2, \tag{3.12}$$

$$Z \leq 2 \cdot R_1 + \frac{4}{3} \cdot R_2, \tag{3.13}$$

$$R_1, R_2 \geq 0. \tag{3.14}$$

The optimal solution to this linear program matches the prior solution. To improve

the solution quality, we could increase the number of rounding points, and thus the number of intervals. In particular, we consider $|D|$ rounding parameters that evenly divide the interval $[1, 2)$. The resulting linear program will have $|D| + 1$ constraints plus the nonnegativity constraints. The first constraint will ensure that the residual cost fractions sum to 1 while the remainder will maximize the minimum value. For a given number of rounding parameters $|D|$, a specific rounding parameter $\delta_k = (|D| + (k-1))/|D| = 1+(k-1)/|D|$, and an interval number $d \leq |D|$, the maximum increase in visitation frequency for nodes in intervals d with $k \leq d$ is $(|D|+d)/(|D|+(k-1))$. For nodes in intervals d for which $k > d$, the maximum increase in visitation frequency is $2 \cdot (|D| + d)/(|D| + (k-1))$. The resulting linear program can be formulated as:

maximize Z (DLP-$|D|$)

subject to

$$\sum_{d=1}^{|D|} R_d = 1, \tag{3.15}$$

$$Z \leq \sum_{d=1}^{k-1} 2 \cdot \frac{|D| + d}{|D| + (k-1)} \cdot R_d + \sum_{d=k}^{|D|} \frac{|D| + d}{|D| + (k-1)} \cdot R_d, \qquad \forall\, k = 1, \ldots, |D|, \tag{3.16}$$

$$R_d \geq 0, \qquad \forall\, d = 1, \ldots, |D|. \tag{3.17}$$

By solving this linear program, we can determine the worst case performance guarantee for any number of evenly spaced rounding parameters. We ran this optimization problem with a number of rounding parameters ranging from 1 to 1000. Table 3.7.1 shows first few objective function values.

For 1000 rounding parameters, the optimal value to the linear program was 1.44321. This mirrors the analytical results from Chapter 2. Figure 3-1 shows the optimal linear programming values for all cases.

As with the worst case example for $|D| = 2$, the optimal solution to the linear

$\|D\|$	Z^*	$\|D\|$	Z^*	$\|D\|$	Z^*
1	2	8	1.50859	15	1.47763
2	1.71429	9	1.50119	16	1.47544
3	1.62162	10	1.49528	17	1.4735
4	1.57598	11	1.49046	18	1.47178
5	1.54886	12	1.48644	19	1.47024
6	1.5309	13	1.48305	20	1.46885
7	1.51813	14	1.48015		

Table 3.1: Optimal linear program solution values for selected discretizations.

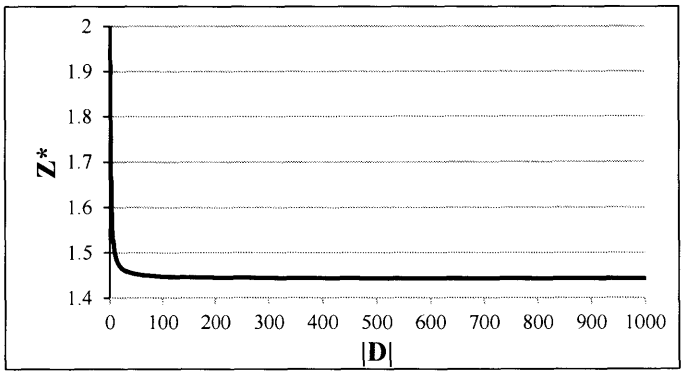

Figure 3-1: Optimal linear programming solution values for discretizations $1, \ldots,$ 1000.

program, namely the R_d values, can be used to construct a tight example, which implies that the analysis is tight for each level of discretization, assuming a uniform discretization is used. In computational tests of the same 200 instances discussed in Section 3.5 but with infinite planning horizons, the a-priori shifted power-of-two algorithm performed much better than the worst case guarantee resulting from the linear programming analysis. Even with only one rounding parameter, namely $\{1\}$, the algorithm yielded solutions that are within 30% of the corresponding infinite horizon lower bound, with the vast majority within 20%. For a larger number of rounding points, this difference drops to 15%, still well below the empirical bound of 44%. In addition to its performance characteristics, the a-priori shifted power-of-two algorithm is operationally attractive due to the cyclic and nested schedules it produces and the fact that the algorithm runs in less than a second for 50 rounding points, which has a worst case guarantee of 1.45.

3.7.2 Ratio Based Rounding Parameters

To this point, via the use of linear programming, we have shown empirically that choosing the best of evenly spaced shifted power-of-two rounding parameters has a limiting worst-case performance ratio close to $1/\ln(2)$. In this section, we will establish a limiting result of $1/\ln(2)$ analytically by judiciously choosing the rounding parameter values (rather than choosing them as equally spaced between 1 and 2). As motivation, consider the prior case with two rounding parameters, $\{1, 3/2\}$. In that case to find the worst-case performance ratio, we needed to solve the following max-min problem:

$$\max_{\substack{R_1,R_2:\\ R_1+R_2=1}} \min\left\{\frac{3}{2}\cdot R_1 + 2\cdot R_2, 2\cdot R_1 + \frac{4}{3}\cdot R_2\right\}$$

The solution $R_1 = 4/7$ and $R_2 = 3/7$ places more residual cost weight on R_1 to exploit the higher frequency ratio of 3/2 as opposed to 4/3. To develop an improved performance bound, suppose instead that we choose the base parameters as 1 and δ so that we balanced the increase in frequency. That is, so that $\delta/2 = 2/\delta$. Then $\delta = \sqrt{2}$

and the optimal solution becomes $R_1 = R_2 = 1/2$ with an optimal max-min value (by substitution) of $1 + \sqrt{2}/2 \approx 1.707$ which improves slightly on $12/7 \approx 1.714$. It is easy to show that no other choice for the base parameter δ gives a better worst-case performance ratio. Note that for the rounding parameters $\{1, \sqrt{2}\}$, the associated linear program becomes:

$$\text{maximize } Z \qquad \text{(DLP-}\sqrt{2}\text{)}$$

subject to

$$R_1 + R_2 = 1, \tag{3.18}$$

$$Z \leq \sqrt{2} \cdot R_1 + 2 \cdot R_2, \tag{3.19}$$

$$Z \leq 2 \cdot R_1 + \sqrt{2} \cdot R_2, \tag{3.20}$$

$$R_1, R_2 \geq 0. \tag{3.21}$$

By symmetry we might argue that the solution is $R_1 = R_2 = 1/2$. To establish that this solution is optimal by duality, note that the objective value of this symmetric solution equals the objective value of the linear programming dual problem with dual variables chosen as $1/2$ for each inequality constraint and $1 + \sqrt{2}$ for the equality constraint.

The choice of rounding parameters $\{1, \sqrt{2}\}$ divides the interval $[1, 2)$ into two regions $[1, \sqrt{2})$ and $[\sqrt{2}, 2)$ whose ratios of right endpoints to left endpoints are equal (i.e., $\sqrt{2}/1 = 2/\sqrt{2}$). To generalize this, suppose we divide the interval $[1, 2)$ into $|D|$ sub-intervals in which the ratio of the right endpoints to the left endpoint is the same for all subintervals. This yields the following system of $|D| - 1$ equations of the rounding points $\{1, \delta_1, \ldots, \delta_{(|D|-1)}\}$:

$$\frac{\delta_1}{1} = \frac{\delta_2}{\delta_1} = \cdots = \frac{\delta_{(|D|-1)}}{\delta_{(|D|-2)}} = \frac{2}{\delta_{(|D|-1)}}. \tag{3.22}$$

Lemma 3.7.1 *A solution to Equations* (3.22) *is* $\delta_q = 2^{q/|D|}$ *for* $q = 1, \ldots, |D| - 1$.

Proof : We will prove the result by showing that the proposed solution yields the

same value for each ratio. Note that each ratio above can be written as $2^{q/|D|}/2^{(q-1)/|D|}$ for some value of $q = 1, \ldots, |D|$ (i.e., $1 = 2^{0/|D|}$ and $2 = 2^{|D|/|D|}$). The value of each ratio is thus $2^{(q/|D|)-((q-1)/|D|)} = 2^{1/|D|}$ which yields the result. ∎

Based on the result from Lemma 3.7.1, the rounding parameters will be $2^{q/|D|}$ for $q = 0, 1, \ldots, |D| - 1$. With these rounding parameters, a generalization of the prior linear programming duality argument shows that the optimal solution is to place an equal fraction, $(1/|D|)$, of the residual cost in each interval. In particular, $R_d = 1/|D|$ for all $d = 1, 2, \ldots, |D|$. For a given number of discretizations, however, it remains to be shown that this choice of rounding parameters is in fact optimal. In particular, does there exists another set of $|D|$ rounding parameters that dominate $2^{q/|D|}$ for $q = 0, 1, \ldots, |D| - 1$?

With the prior choice of rounding parameters and the resulting residual cost assignments, the maximum residual cost increase for each rounding parameter will be the same. For the rounding parameter 1 it becomes $(1/|D|) \cdot (2^{1/|D|} + 2^{2/|D|} + 2^{3/|D|} + \cdots + 2^{|D|/|D|})$. By increasing the number of intervals to infinity, we obtain:

$$\lim_{|D|\to\infty} \frac{1}{|D|} \cdot \sum_{d=1}^{|D|} 2^{d/|D|} = \lim_{|D|\to\infty} \frac{1}{|D|} \cdot \int_{d=1}^{|D|} 2^{d/|D|} \mathrm{d}d$$
$$= \lim_{|D|\to\infty} \left(\frac{2}{\ln(2)} - \frac{2^{1/|D|}}{\ln(2)} \right) = \frac{2}{\ln(2)} - \frac{1}{\ln(2)} = \frac{1}{\ln(2)}.$$

This calculation establishes the following result.

Theorem 3.7.2 *Suppose we choose the best visitation schedule of $|D|$ a-priori shifted power-of-two heuristic solutions, with the rounding parameters chosen as $2^{q/|D|}$ for $q = 0, 1, \ldots, |D| - 1$. Then, as $|D|$ tends to infinity, the worst-case performance ratio is $1/\ln(2)$.*

Similar to the analysis of the evenly spaced rounding parameters, the linear program solution yields not only the performance guarantee, it also provides the worst case instance that proves the bound is tight. In particular, the instance that equally distributes the residual costs among the intervals and has cycle limits sufficiently

close to the rounding parameters. As might be expected from the prior case of $|D| = 2$, computational testing has shown that the difference between choosing the $2^{q/|D|}$ rounding parameters and choosing the rounding parameters equally spaced over the interval $[1, 2)$ is minimal. The largest discrepancy in optimal objective values with up to 1000 base parameters was only 0.00718.

3.8 Conclusions

In this chapter we proposed a graph visiting problem and examined associated optimization related models and issues. We provided a general mixed integer programming formulation, as well as an improved cyclic formulation. We showed that for a special case in which the frequencies are powers of 2, the linear programming relaxations of both formulations have integral optimal solutions. Using this fact, we showed that the integrality gap of both formulations is 2 and that the optimal values for the general and cyclic MIPs differ by at most a factor of 2. Finally, we examined a linear programming formulation that provides the worst case performance guarantee of a cyclic rounding heuristic when the costs are submodular. Using this formulation we show that a so-called a-priori shifted power-of-two algorithm has a worst case performance of $1/\ln(2) \approx 1.4427$ for a sufficiently large set of rounding parameters. In addition to showing the worst case guarantee, the linear program provides the worst case instances that attain the bound for each level of discretization. We conjecture that the ratio based rounding parameters are the optimal choice of a-priori rounding parameters.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Modular Maintenance and System Assembly

As discussed previously, many modern systems are modularly designed, which allows maintenance to be performed on individual modules without performing maintenance on other parts of the system. Selective maintenance is often achieved by removing a failed module and performing the needed maintenance in a facility that specializes in the repair of a module or a small subset of modules. To accomplish this task, other modules in the system must be disconnected from the failed module but need not be taken apart to a significant degree. Such a design is advantageous from a maintenance perspective since specialized manpower and equipment needed to repair modules can be placed at a centralized location while the module's removal and replacement can occur at, or very close to, the operational location for each system. However, to return the system to an operational state, operating locations must either wait for modules to be repaired and returned to them or must hold an inventory of modules on-site to replace the failed module. This results in a tradeoff between centralized maintenance capacity, transportation times, and inventory levels of modules at the operating locations. It is this tradeoff that is the focus of this chapter. Furthermore, we will concentrate on a specific environment, the modularly designed engines found in aircraft operated by the Air Force.

The Air Force performs preventative maintenance during scheduled aircraft down-

times so as to prevent a failure during use. Jet engines on fighter aircraft, for example, are maintained in this manner since an in-flight engine failure can have disastrous consequences. Turbines used in commercial power plants are maintained in a similar manner. During evenings, weekends, or other periods of low electrical usage, preventative maintenance can be performed to ensure sufficient generating capacity is available during peak usage times. In these situations, we must decide when to perform maintenance on each module and the extent of the maintenance to perform. While the decision of when to perform maintenance is similar across many different types of systems, the amount of maintenance to perform and its relative impact is specific to individual module types. For instance, metallic components on an aircraft engine are susceptible to cracking due to the high vibration environment. Once a metallic component has a crack of a predetermined length, it must refurbished or replaced with a new component. Replacement is not the only way to extend the life of the module that contains the metallic component. Visual inspection can verify that no large cracks exist and thus the module can continue to be used for a certain amount of time. More in-depth, non-destructive methods such as dye penetration, x-ray, or ultrasonic testing can extend the usable life of the module even further. However, to obtain the longer usable life, more maintenance manpower and equipment must be used to perform these tests and potential fix any discrepancies that are found. Accordingly, the level of module maintenance ties directly back to the tradeoff among maintenance capacity, transportation times, and inventory levels.

4.1 Background

In an Air Force context, the most central modular systems in terms of cost and operational criticality are jet engines flown on fighter aircraft. As discussed in Section 1.1, these engines support a significant portion of the Air Force's combat capability. Much like the modular systems discussed in Chapters 2 and 3, these jet engines require preventative maintenance on a number of critical components. After a given number of flight hours, some subset of the modules in the engine will need to be

maintained. Due to the modular design of the engines, maintenance activities can occur at the base where the aircraft are flown or the modules can be transported to a centralized maintenance facility. Currently, the decision of where to repair a module is based mostly on the level of repair required. Complex repairs are performed at the depot while more simple repairs are performed in on-base maintenance shops. The U.S. Air Force currently has 3 depots located at Ogden Air Logistics Center (OO-ALC) in Utah, Oklahoma City Air Logistics Center (OC-ALC), and Warner-Robins Air Logistics Center (WR-ALC) in Georgia. Fighter engine repair work is performed primarily at OC-ALC which is not located near major fighter bases. The decision to perform this work at OC-ALC is based on a long history that entails technical, political, and geographic considerations.

Module maintenance capacities are not the only maintenance limitations faced the Air Force. Often there are also limits on the number of systems that can be returned to an operational state at any point in time. For aircraft engines, this limitation could be imposed by the stands needed to test-fire the engines prior to installation on an aircraft. This capability, unlike the module maintenance capability, is usually placed on or near the base of operations as transporting fully assembled engines is costly and risky. In general, there is a range of manpower and equipment related capacities that must be considered as part of the module maintenance and engine assembly and testing.

Accordingly, the decision between on-site and centralized maintenance relies heavily upon a robust transportation network and sufficient inventory. For bases in the United States, such a transportation network is in place through both military transportation and commercial shippers. However, forward deployed bases have much less robust transportation links. A natural way to address transportation concerns is to hold more inventory (engines and modules) at these forward locations. These austere bases pose an additional problem when considering storing inventory. Due to their locations, they are limited in size due to political and security considerations. For instance, holding inventory at these forward locations requires a warehouse which in turn requires security manpower. The additional security manpower requires ad-

ditional support functions (i.e., dining facilities, sleeping quarters, etc.). The same can be said of adding additional maintenance capability at these austere locations. Thus, a natural tension arises between the need for on-site inventory, on-site repair capability, and a robust transportation network.

While the technical aspects of engine maintenance are quite complex, the problem is further complicated by organizational issues within the Air Force. In particular, the engines we consider are flown primarily by aircraft owned by Air Combat Command (ACC) or one of the geographic commands (U.S. Air Forces in Europe, Pacific Air Forces, etc.). These organizations also oversee the on-site maintenance capabilities resident in the Aircraft Maintenance Units (AMUs) in the local bases. While the AMUs report to the combat portion of the Air Force, the depots are part of Air Force Materiel Command (AFMC) which is tasked with acquisition and sustainment. The transportation links between the operational bases and the depots fall under the purview of Air Mobility Command (AMC) and the associated joint United States Transportation Command (TRANSCOM). Finally, a significant portion of the inventory needed for preventative maintenance is acquired by the Defense Logistics Agency, a Department of Defense level organization which supplies more than 80 percent of the militarys spare parts and nearly 100 percent of its consumable items [1]. Through the model we develop in subsequent sections, we will show the interaction between different organizations in both planning and execution. For instance, if there are repair capacity changes at an AFMC depot, the need for on-base repair, transportation, and inventory will be affected. By jointly considering the decisions of all organizations involved in the engine supply chain, there is a potential for improved support and cost savings.

In the remainder of this chapter we will develop a modeling framework for use in maintenance scheduling for a number of modular systems used and maintained at numerous locations. The model captures not only the high level planning aspects of interest to policy makers; it also yields detailed insight for schedulers and maintainers to use in their daily operations. While there is inherent randomness in the maintenance environments we seek to model, we will initially model the problem

deterministically and include a predetermined level of safety stock to ensure that operations can continue even if a relatively small number of engines experience premature failure. This method for dealing with randomness is strongly grounded in the fact that with a large number of independent factors affecting the presence of uncertainty over a long period of time, safety stock can effectively mitigate the outcomes. However, this method is not effective when considering day-to-day or week-to-week scheduling operations. For this reason, we will further consider stochastic events (modules that degrade stochastically) as part of the development of the specialized operational models and algorithms in the following chapter.

The intent of the models we develop is to allow policy makers to make tradeoffs between maintenance resources, transportation capacity, inventory, and operations tempo and then evaluate the consequences of those decisions. While costs have been the focus of these decisions in other models, we take an operational viewpoint. By doing so, we also allow maintenance schedulers to use the same framework to make tactical and operational decisions about engine assembly and module repair. This permits direct traceability between high level policy decisions and operational impacts. In addition to high level policy decisions, these models also allow maintenance schedulers to not only schedule preventative maintenance, they also allow them to make decisions about surge maintenance capacity and other near term tradeoffs.

4.2 Assumptions and Modeling Approach

In order to model the module repair and engine assembly problem, we will consider four distinct types of inventories at each operating location: reparable modules, serviceable modules, reparable systems, and serviceable systems. These high level inventories include separate inventories for each reparable and serviceable module type.

- Reparable modules are those modules that have been removed from systems but have not yet undergone maintenance to extend their usable lives. Modules in the reparable inventory are characterized by their remaining usable life which

may vary significantly across modules of a specific type.

- Serviceable modules are reparable modules that have undergone maintenance and are available for system assembly. Serviceable modules are also characterized by their remaining usable lives.
- A reparable system is a figurative shell into which serviceable modules can be installed to create a serviceable system
- Serviceable systems are used to meet external demand and are comprised of one module of each type which together define the system's usable life.

Preventative maintenance can blur the distinction between reparable and serviceable modules. Namely, a reparable module that has a large remaining life might be placed directly into the serviceable inventory without undergoing maintenance. In the model we develop in this chapter, we assume all modules that are removed from a system first pass through the reparable inventory before entering the serviceable inventory. If no repair is performed on a module, this shift from the reparable to serviceable module inventory occurs with little or no lead time. As a consequence, this assumption has no affect on the applicability of the model. While the distinction between reparable and serviceable modules might not be clear, there is a well defined delineation between reparable and serviceable systems.

In this work, we think of a reparable system as a figurative shell into which serviceable modules can be installed to create a serviceable system. The figurative shell might be a physical shell into which modules are installed or it might be nothing more than a serial number that is assigned to the complete system once it is assembled. As a consequence of our definition of a reparable system, we are assuming that a system in need of repair is completely disassembled and all modules from that system are placed in the reparable module inventory. This allows each module to be considered for maintenance and also allows us to track system and module inventory in aggregate, rather than as individual entities tied to a specific physical item. By doing so, the model we develop subsequently for jointly managing maintenance, transportation, and inventory is reduced in size and simplifies the corresponding optimization

problem substantially. We note that in an operational setting, systems would not be fully disassembled. Rather, only the modules to be repaired would be removed. Since our goal in this chapter is to develop a model that addresses long term maintenance, inventory, and transportation policies, such an assumption is valid.

Serviceable systems are those systems used to meet external demand. These serviceable systems are comprised of one module of each type and together these modules define the system's usable life. Each module in the system has a remaining usable life and, in this work, we consider systems in which the module with the minimum usable life defines the system's usable life. Implicitly, this means that each module is independent of every other module. While this might not be the case when modules are close to failure (i.e., excessive vibration in one module might negatively impact another module), in the systems we consider, the end of the usable life of a module is not the failure point of that module but rather is the point at which the risk of failure outweighs the benefit from further usage of the module. In addition, due to the conservative nature of the usable limits, and the complexity of the resulting models, we do not consider random failures directly. Thus, we assume that each module could continue to function effectively until the usable life is reduced to zero. This is not a restrictive assumption as this is the current operating procedure for aircraft engines since an engine failure can result in the loss of an aircraft and possibly the loss of a pilot. In contrast, less critical aircraft systems, especially those that have a constant failure rate, are maintained when needed (i.e., after a failure or significant degradation has occurred).

Based on our definition of a reparable system, the decision assemble a serviceable system results in a full reassembly using serviceable module inventory. Once the system has been fully reassembled, it must undergo testing prior to use. We will model limited assembly and testing capability jointly. Due to the relatively short leadtimes required to move an engine through assembly and testing, we will constrain the inflow of systems into the assembly and test process. We assume that this capacity (mostly manpower and test stand related) is not impacted by the remaining life on the engine but rather is a physical constraint based on the number of test stands, as

well as the required setup and teardown times.

We note, however, that modules might never reach the zero state since preventative maintenance actions could ensure that the module is always serviceable. Opportunistic maintenance could be carried out to balance the maintenance workload over time. For the systems we consider, the usable lives are quite long. Hence, in our model, we discretize the remaining life into intervals to obtain a reasonable number of module states and thus system states. The discretized intervals, however, need not be of the same length and might be coarse for higher remaining serviceable states and refined for lower states. For example, if the maximum life of an engine before required removal is 5000 operating hours, with module lifetimes of a similar magnitude, we may define these intervals as 0 to 100 hours, 100 to 500 hours, 500 to 1000 hours, 1000 to 2500 hours, and 2500 to 5000 hours. We define the state of a system (module) to be the amount of its usable life remaining (in days, flight hours, etc.) until it must be maintained in some way. Another important aspect of the problem closely related to the usable life is the usage rate of systems. In Section 4.3.2 we discuss more details about how the usage rates are developed. In addition, the Air Force develops Aircraft Standard Utilization Rates as part of the Flying Hours Program [4, 5].

4.2.1 Single Location Model

At an individual operating location (a fighter aircraft base in the Air Force context), the maintenance process is primarily driven by systems reaching the end of their usable lives and thus requiring maintenance. Since the component with the minimum usable life defines the usable life for the system, the system reaching the end of its usable life implies that at least one module in that system has reached the end of its usable life. When the system is disassembled and placed in the reparable systems inventory, the modules from that system are placed in the appropriate reparable module inventories. The next decision to be made is how many of each module type, with specific hours of usable life remaining, to enter into their respective maintenance processes and the level of usable life these modules should have following the completion of the repair activity. This decision must be made for each feasible combination of

the starting and ending states (i.e., since maintenance can only increase the number of usable hours, the ending state can be no less than the starting state). For instance, if there are two modules of the same type both with 100 hours of remaining life, we might choose not to maintain one of them and move it directly to the serviceable module inventory. The other module, however, could be entered into major maintenance thus increasing the usable life from 100 hours to 500 hours with an appropriate maintenance delay. While in repair, this second module will consume maintenance resources that will limit the number of other modules of that type that can be repaired. In an Air Force context, module repair will last between a week and a few months, depending upon the level of maintenance required.

Modules that have finished repair are moved into the serviceable module inventory where they remain until removed for installation on a system. System assembly is the next decision that must be made, namely how many systems to enter into the assembly and test sequence. For maintenance continuity and capacity considerations, the number of systems entering maintenance might be restricted both from above and below. The lower bound is specifically aimed at maintenance manpower which, to remain proficient and efficiently utilized, must perform maintenance activities at regular intervals. The upper bound is a direct result of limited maintenance resources, which could be manpower, assembly equipment, test equipment, or a combination thereof. As a serviceable system is made up of one module of each type, the assembly decision entails the matching of modules from the serviceable inventory with a system from the reparable inventory. Once matched with a reparable system, the usable lives of these modules will determine the usable life of the system they form. Once the serviceable modules and reparable system are assembled, they are tested and the system is returned to the serviceable system inventory. Assembly and test might require multiple periods to complete (at most a few weeks in the case of jet engines), resulting in a delay in returning the system to the serviceable system inventory. We assume all systems that enter assembly successfully complete testing and return to the serviceable inventory without requiring further maintenance. While some engines do not pass the test fire, these events are rare and can usually be addressed quite

quickly.

At the operating locations, demand for systems is realized and systems in the serviceable inventory are used to meet this demand. We assume that demands are known ahead of time which is reasonable considering the Air Force flying hours program [4, 5]. Our overarching goal is to provide enough serviceable systems to meet the demand at every time period. Recognizing that the deterministic demand and no random failure assumptions are restrictive, we add a buffer of serviceable systems above and beyond the known demand to help mitigate the affects of unforseen events. Keeping a large inventory of serviceable systems, however, is not desirable due to storage limitations, security requirements, and other considerations. Accordingly, our objective will be to maintain the serviceable systems buffer at each time period with a penalty for not meeting this target but no benefit for being over.

Based on this description of a single operating location, we define six events that occur in each time period. The flow of inventory, modules and systems, both reparable and serviceable, that we will use in the mathematical model relies upon the order of these events as they occur in a time period. Figure 4-1 illustrates the cyclic nature of these events for a single operating location in which a system consists of four different modules.

1. Systems that have reached the end of their usable lives, and their corresponding modules, enter their respective reparable inventories.
2. Modules coming out of maintenance enter the serviceable inventory in the appropriate state.
3. **Decision:** Modules are removed from the reparable inventory and entered into the repair process.
4. **Decision:** Systems are removed from the reparable inventory (backlogged systems awaiting repair) and modules from the serviceable inventory to assemble a serviceable system.
5. Systems that have completed assembly and test are returned to the serviceable inventory.

6. System demand is realized and the usable life for each system is decremented accordingly.

The number of each module to repair (item 3) and the number systems to assemble (item 4) are the decisions to be made in each time period.

As the model is intended for an operational context, we envision that it will be solved in a rolling horizon manner. Accordingly, at the beginning of the planning horizon there will be systems in use, systems in assembly, and modules in repair that we must account for in our model. During the planning horizon, we do not make decisions about these systems and modules but will flow through the system in conjunction with the systems and modules for which we will make maintenance and assembly decisions going forward. The initial flow of these systems and modules into our model will be captured through data inputs to the model.

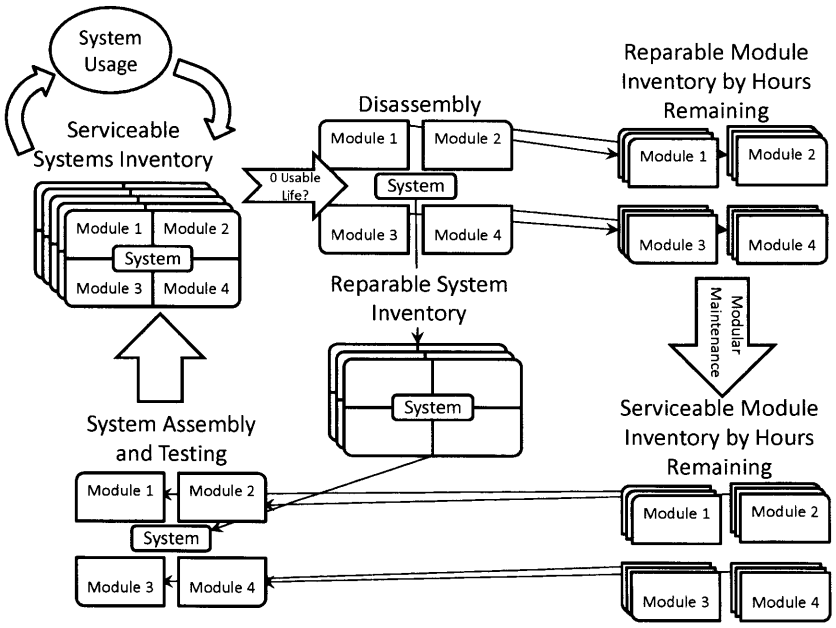

Figure 4-1: Process Flow at a Single Operating Location

4.2.2 Depot Location Model

When there are a number of operating locations, such as Air Force bases, system usage and maintenance capacity might vary between locations and thus it may be advantageous to transport modules and systems between locations for repair, assembly, and test. In addition to the operating locations, the specialized skills and equipment needed to perform modular maintenance may necessitate the creation of a centralized maintenance facility, which we call a depot, that would perform module repair. We assume no systems operate at the depot, however. The motivation behind a depot is to have a shared repair facility that can serve the operating locations. With such a location, centralized maintenance resources can be used to balance the maintenance demands on the operating locations. In addition to performing maintenance, the depot can hold inventory of serviceable modules that can be used to resupply operating locations. The purpose of the depot is to model the tradeoff between centralized versus distributed maintenance.

The depot serves as a maintenance hub and is modeled in the same manner as the maintenance portion of an operating location. However, as the depot does not operate the system and thus, by assumption, does not disassemble systems, the incoming inventory of reparable modules is based solely upon shipments from operating locations. As with the operating locations, we assume that each module type is repaired in a separate capacity constrained maintenance shop at the depot, each of which operates independently of the other maintenance shops. The number of periods to increase the usable life of a module from one state to another is a key decision made for each module at the depot. These repair times at the depot depend only upon the module type, beginning state, ending state, and possibly the period in which repair is initiated. Maintenance capacity limits the total number of modules of a given type that can be in maintenance at a specific time period. Figure 4-2 illustrates the depot process, again for a four module system. In this work we assume the depot does not perform system assembly and test but our formulation can easily be extended to allow such activities to occur. In an Air Force context, the OC-ALC performs the

majority of in-depth module repair and a small portion of engine assembly/test.

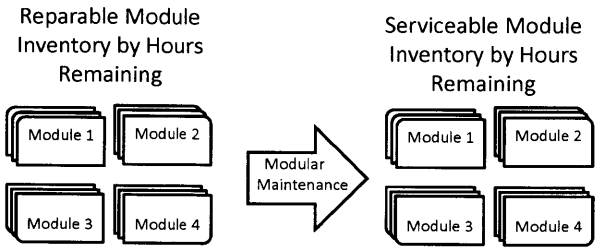

Figure 4-2: Process Flow at Depot Location

4.2.3 Multiple Locations and Capacity Constrained Transportation

While the single location model is challenging and of interest, expanding our scope to multiple locations is a better representation of reality. At a single location, the tradeoffs that can be made are solely between maintenance capacities and hardware inventories. By considering multiple locations, we add to these tradeoffs by including limited transportation resources in the model. Allowing systems and modules to flow between locations facilitates load balancing between locations and maintenance pooling at a depot. By varying the inventories, transportation assets, and maintenance capacities, it is possible to understand the specific tradeoffs and how they affect the entire operational environment. For this work we assume maintenance resources are not transported between locations, only reparable and serviceable systems and modules can be transported. From a practical standpoint, we limit the transport of serviceable systems to occur only when coming out of assembly and test. Another, more restrictive assumption that could be made, is to disallow the transport of serviceable systems. Either assumption is valid in many cases as transporting a serviceable system posses an inherent risk. Transporting a serviceable system multiple times without retesting the system is a risk not likely to be taken.

Transportation of inventory between locations is characterized by both the time needed to travel between locations as well as the capacity available between locations.

Both of these values depend upon the resources assigned to a transportation link. While the capacity between locations might vary on a period by period basis, the travel time between locations is relatively static for the scope of problems we consider. Extending the mathematical model we propose to allow for varying travel times is straightforward. While a metric assumption (satisfying the triangle inequality) on travel times is not required for our model, in many instances, such an assumption is valid.

With the combination of multiple locations and transportation between these locations, we now define eight events that occur each time period in the full model. In Figure 4-3 we illustrate these events for a single depot, three operating locations, and a four module system. As illustrated in the figure, throughout this work we will consider a single depot with multiple operating locations arranged in a classic two-echelon framework. By appropriately redefining set notation, the mathematical model as written can accommodate multiple depots and/or more than two echelons. The sequence of events is as follows:

1. Systems that have reached the end of their usable lives, and their corresponding modules, enter their respective reparable inventories at each location.
2. Systems and modules (reparable and serviceable) in transit from other locations arrive at their destinations and enter their respective inventories.
3. Modules coming out of maintenance enter the serviceable inventory at their respective locations.
4. **Decision:** Modules are removed from the reparable inventory and entered into the repair process at their respective locations.
5. **Decision:** Systems are removed from the reparable inventory (systems awaiting repair) and modules from the serviceable inventory to assemble a serviceable system at their respective locations.
6. Systems completing assembly and test are returned to the serviceable inventory at their respective locations.

7. **Decision:** Systems and modules (reparable and serviceable) are removed from their corresponding inventories and shipped to other locations.
8. System demand is realized and the usable life for each system is decremented accordingly.

The number of each module to repair (item 4), the number systems to assemble (item 5), and transhipment decisions (item 7) are the decisions to be made in each time period.

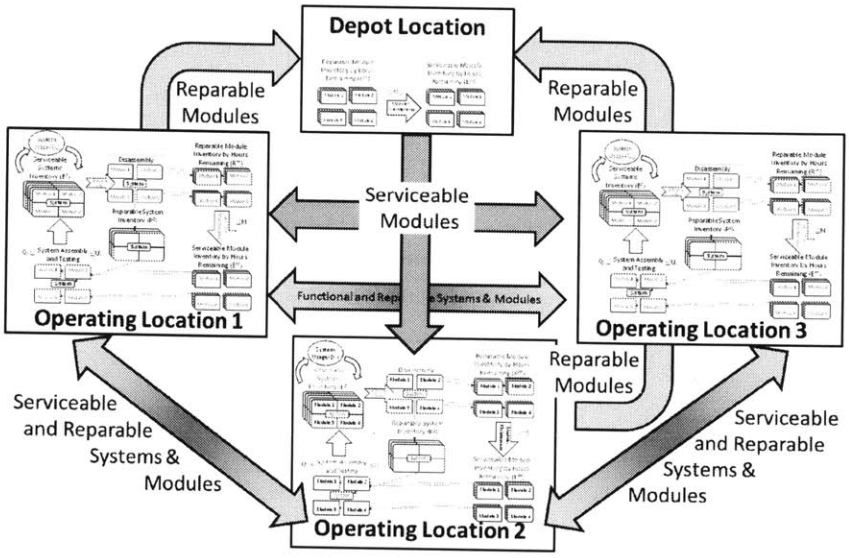

Figure 4-3: Full Maintenance Operation with Operating Locations, Depot, and Transshipment

4.2.4 Combined Model

While the movement of modules and systems allows for load balancing across locations, another important aspect of this problem is load balancing across time. Having a large standing maintenance capability that can react to surges in maintenance demand is extraordinarily expensive in both equipment and manpower. Unlike commercial settings where short-term maintenance contracts could be executed, in military environments, manpower cannot be increased quickly due to the extensive training required to work on unique, complex systems. Accordingly, we seek to balance and

reduce the maintenance workload that is required over time while still meeting mission requirements. These strategic capacity decisions are a key part of the tradeoffs that can be explored with the combined model.

To this point, we have described maintenance and transportation resources as hard constraints. They can, however, be viewed as soft constraints with additional capacity incurring some cost. However, it can be difficult to estimate these costs accurately and it can be difficult to translate shortages of operational systems into costs. Accordingly, throughout our model and algorithm development our focus will remain on meeting the demands of the operational community. Rather than seeking to minimize monetary costs subject to specified operational readiness rates (ORR) or maximizing ORR subject to budget constraints, we strive to use an operational objective (meeting the aircraft demands from the flying hours program) and numerous operational and logical constraints. One reason for doing this is the inherent difficulty in monetizing assets in certain contexts. For instance, for assets such as maintenance facilities, what is the appropriate value for the facility and how should those costs be amortized between organizations and over time. Secondly, by focusing on operational considerations, this framework can help bridge the gap between policy and practice. When used in a policy setting, decision makers can use their intuition and insight to modify the resource allocations subject to the manpower and budgetary environments they face. Operational planners can use the same model, with the same assumptions and inputs, to schedule maintenance. This common framework also allows policy makers and practitioners to have a common success metric rather than having one organization focus on dollar amounts and another focus on meeting operational demand. To this end, our model will pose four fundamental tradeoffs that impact policy, planning, and operations:

- resource pooling of both maintenance capacity and inventory versus transportation capacity,
- in-depth maintenance practices with long processing times (possibly necessitating higher inventory levels) versus lower level maintenance with quicker turn

around times,

- the impact of cannibalization in which serviceable modules are pulled from a non mission capable system in order to repair another system, and
- module matching in the system assembly process to avoid large changes in the demand for maintenance capacity.

4.2.5 Solution Methodologies

We will first formulate the full Modular Maintenance and System Assembly problem as an integer programming problem. Due to the complexities discussed earlier, this formulation is theoretically and practically intractable to solve even for small instances. Accordingly, in Chapter 5 we consider a hierarchical solution methodology. Beyond the intractability of the full formulation, we consider the decomposition approach due to inherent randomness that will occur in the actual problem. Incorporating random failures in the full model would exacerbate the computational issues. Also, based on discussion in Section 4.1, the different organizations involved in decision making is a third reason for the decomposition. Finally, there are different planning horizons that are of interest when considering module repair and system assembly. In particular, the effects of system assembly decisions arise much sooner than those of module repair.

As part of this decomposition approach we will consider three types of models, (i) the full *planning model* that is solved infrequently over a long horizon with a relatively coarse time increment and continuous variables, (ii) an integral *system assembly model* solved frequently and over a shorter time horizon, and (iii) integral *module repair models* (one for each module type) that are also solved frequently and over a planning horizon that is longer than the horizon for the system model but shorter than the full planning model's horizon. The decomposition approach leverages the long horizon of the planning model to make near term decisions in the system assembly and module repair submodels. By using this information, long term impacts of current decisions can be accounted for and mitigated. These long term impacts are captured through

serviceable system and serviceable module target inventories that will be passed from the planning model to the system assembly and module repair submodels. The full description of the decomposition is detailed in the beginning of Chapter 5. In addition to the decomposition approach, we will also develop algorithms that can be used to solve special cases of the system assembly submodel. To account for the inherent randomness in the problem, we also generalize the module repair submodel to include situations in which modules degrade stochastically over time.

The remainder of this chapter is organized as follows. In the next section we will develop our full mathematical model. Following the development of the mathematical programming formulation, we will explore how the formulation captures the tradeoffs mentioned previously and how planners and schedulers can use the resulting solutions when faced with these tradeoffs. Finally, we will review the literature relevant to the model and identify the differences between the model we address and previous work. In the next chapter, we develop a decomposition approach that yields actionable solutions for policy makers while also allowing the operational community to use the same planning framework to plan and execute maintenance schedules. In the final portion of that chapter, we will also exploit ideas presented in Chapters 2 and 3, specifically the concept of nestedness and show how early consideration of maintenance concerns can drastically reduce the scheduling complexity and life-cycle maintenance costs.

4.3 Full Mathematical Model

To this point we have described our assumptions concerning the way the maintenance environment operates during each period. We now define the mathematical model for this modular maintenance and system assembly problem. We begin by defining the sets used in the model, the data required, and the decision variables. Using these, we will then motivate, describe, and formulate the objective function and individual constraints for the model. Finally, we will combine the objective function and constraints to form the full model.

4.3.1 Sets

In the Modular Maintenance and System Assembly Model we assume a finite planning horizon consisting of T time periods over which maintenance, transportation, and assembly decisions will be made. We define $\mathcal{I}$ to be the set of module types with with module 0 defined as the system. We assume a two-echelon maintenance and operational environment with $\mathcal{L}$ denoting the set of locations and location 0 denoting a depot facility. The depot can perform maintenance on modules but does not perform system assembly and does not operate the system. While we delineate the depot as a special location, it might in fact be physically collocated with an operational location. The delineation is important so that we can manage the depot's maintenance resources separately from the operating location's maintenance resources.

Based on the assumptions of module independence and the notion of remaining usable life, we define the sets $\mathcal{M}$ and $\mathcal{N}$ to be the module and system states, respectively. Let $m(i)$ represents the usable life of the module of type i installed on the system. Since the module with the minimum remaining usable life limits the maximum possible usable life of the system, we know that the usable life for a newly assembled system is limited by the minimum usable life for the modules installed in that system, specifically $n \leq \min_{i \in \mathcal{I}} [m(i)]$. This implies that the maximum element in $\mathcal{N}$ might be smaller than the maximum element in $\mathcal{M}$ as not all modules may be able to reach all module maintenance states. That is, some modules types may have maximum usable lives that are shorter than others.

For example, the set $\mathcal{M}$ could be all integers in the interval $[0, 100]$. If one of the module types has a maximum usable life of 75, then the set $\mathcal{N}$ would be the integers in $[0, 75]$. This example of $\mathcal{M}$ and $\mathcal{N}$ accounts for the exact number of usable hours remaining but leads to a very large state space. While covering the same range of usable life, the set $\mathcal{M}$ could be aggregated, for example, $\{0, [1, \ldots, 10], [11, \ldots, 20], [21, \ldots, 40], [41, \ldots, 70], [71, \ldots, 100]\}$ with $\mathcal{N}$ defined in a similar manner. This second representation with bucketed usable life leads to a much smaller state space while still capturing the essential regimes.

Obvious additional sets we could define are the set of systems and the set of modules by type, that is, sets to track each system and module individually. In an operational setting, each system and module would be tracked individually for accountability purposes. However, this model serves as a planning tool to allow policy makers to consider long term tradeoffs between maintenance capacity, transportation resources, and inventory. Over longer planning horizons, tracking each system and each module individually increases the size of the model and necessitates binary decisions which complicate the solution methodology. Thus, we consider system and modules in an aggregate sense at each location rather than tracking them individually. The aggregation relies on the fact that many modules or systems may be in the same state at the same location. We track the aggregate number of systems or modules in a given state and location at a point in time. We do not differentiate between systems and between modules that are in same state.

This aggregation leads to a pitfall when combined with the transportation of serviceable systems. Namely, when a serviceable system is shipped between locations, we do not track the states of the individual modules in that system. When the system comes back for maintenance, at least one module has no remaining usable life. However, which module that is and the remaining life on the other modules in the system is unknown. As a conservative assumption, we will assume that when a system is shipped from location l to location l' and comes in for service at location l', all modules in the system have no usable life remaining. In essence, this implies all modules installed on serviceable systems that are transported have the same remaining usable life. Such an assumption disincentivizes the transport of serviceable systems and results in higher maintenance resource usage. By expanding the formulation to track the shipment of systems in each state, this assumption could be removed. However, this would increase the size and complexity of the formulation. As mentioned previously, transporting serviceable engines posses an inherent risk thus disincentivizing such transports more accurately models reality.

4.3.2 Data

To initialize the inventories of serviceable and reparable systems at each location at the beginning of the planning horizon, we define S^0_{l0} and R^0_{l0} to be the number of serviceable systems available at location l and the number of reparable systems available at location l, respectively. The superscript 0 indicates the system (other superscript indices will refer to modules) while the subscript 0 indicates time period 0. To capture systems currently in assembly and test, we define B^0_{lt} to be the number of systems that will return into service at location l at time t. The decision to assemble and test these systems was made prior to the current planning horizon and due to the time required to assemble and test the system, they will enter the serviceable inventory during the planning horizon. Note that these systems are not denoted by state thus we can not directly track when they will leave the serviceable inventory. Their departure (if it happens during the planning horizon) is captured in the next data element.

Systems currently in use or for which assembly and test or transit decisions were made prior to the first time period might come out of service during the planning horizon. The systems that will leave the serviceable inventory are captured in the data element A^0_{lt} which we define as the number of systems that will come out of service, and need to be disassembled, at location l during time period t. As these systems were assembled prior to the planning horizon, we do not have specific information about the modules contained in these systems. Accordingly, this data is captured in the module data element A^{im}_{lt} that is described later.

As mission execution is our primary objective, we let D_{lt} be the demand for serviceable systems at location l at period t. Let U_{lt} be the upper bound on the number of systems entering assembly and test in period t at location l. For maintenance leveling and proficiency purposes, we assume a minimum number of systems entering assembly and test denoting this quantity by Q_{lt}. For systems in a specific state n, the minimum proficiency level is denoted by Q^n_{lt}.

System assembly and test, results in a system being placed into the serviceable

inventory. We assume this process takes a specified number of time periods based on the location and possibly the time period assembly begins, denoted as γ_{lt}. Once in use, a system can be used to service demand until its usable life has been exhausted. While the number of periods a system can be used until its usable life is depleted depends upon the demand, the number of serviceable systems, and the life remaining on those systems, an estimate can be made based on the historical data or model based forecasts. In an Air Force context, the demand rate is directly related to the planned flying hours program [4, 5]. In addition, recent work has been done that looks at flight hours and sortie assignment for aircraft in order to meet operational objectives [13]. By using this model or other standard Air Force methodologies, the approximate number of days an engine will be in service prior to needing maintenance. Accordingly, we define ψ_{lt}^{n} as the approximate number of periods it will take to exhaust the usable life of a system that was assembled in state n and used at location l if it is placed in service during time period t.

While we do not explicitly model random failures, we recognize that such failures do occur. When a random failure occurs, a prudent course of action is to remove the system from use until the failed module(s) can be repaired as well as a system level check completed to ensure other modules were not significantly impacted by the failure. To account for these random failures, we set a desired threshold of serviceable systems we plan to hold in inventory that can be used in place of failed systems. The number of systems held in inventory explicitly for premature failure mitigation would most likely be based on past failure data as well as current demand. It could also include mission criticality at a location and the transit time to get additional systems to a location. Accordingly, we define ξ_{lt} as the number of reserve systems desired at location l at period t. We assume that an appropriate model that accounts for this uncertainty is used to set the ξ_{lt} values.

Similar to the system quantities, we define S_{l0}^{im} and R_{l0}^{im} to be the initial inventories of serviceable and reparable modules of type i in state m at location l. Systems for which maintenance decisions were made prior to the first period (both those in the field at $t = 0$ and those that will enter service after the first period) might return

for maintenance during the planning horizon along with the modules contained in those systems. This return of reparable modules at period t is captured in the input A_{lt}^{im} defined as the number of reparable modules of type i in state m that will return to the reparable inventory at location l in time period t. In addition, there may be modules in repair at the beginning of the planning horizon. These modules will return to the serviceable inventory at some point in the planning horizon. Let B_{lt}^{im} be the number of modules of type i that will return to the serviceable inventory in state m at location l in time period t.

Based on the modular design of the system, we assume that each module is repaired by a separate maintenance shop that has a capacity that is independent of the other maintenance shops. Each period a module is in maintenance, it uses one unit of the available capacity, M_{lt}^{i}, which is the total available repair capacity at location l for module type i at period t. A module that enters repair must spend a certain number of periods in repair based on the beginning and ending states, the location, and time entering repair. This quantity is denoted as $\tau_{lt}^{imm'}$, the number of periods a single module of type i requires maintenance to increase its state from m to m' at location l given that repair began in period t. In addition to the usage based modules commonly thought of in maintenance repair, we will also consider calendar driven events such as inspections that must be carried out after a certain number of days, independent of the amount of system usage. Such events will be captured through artificial modules that have only one allowable installation state m_i which represents the number of periods between required maintenance. As these artificial modules do not require repair in the standard sense, $\tau_{lt}^{imm_i} = 0$ for all such calendar driven modules. The time required to perform these activities will be included in the system assembly times described above.

When considering multiple locations, we define capacities for transport as well as transit times between locations. The aggregate capacity available to transport systems and modules between locations l and l' at time period t, which we denote by $\mu_{ll't}$, affects where repair jobs occur, as does the transit time between locations, $\nu_{ll'}$. To allow for more refined decision making, we define serviceable and reparable

transport capacities for each module (including the system with $i = 0$) as $\mu^{is}_{ll't}$ and $\mu^{ir}_{ll't}$, respectively. In many instances, these limits would be set to either 0 or ∞ to indicate no movement or free movement of specific module types. At the beginning of the planning horizon, some systems and modules might currently be in transit between locations. Accordingly, we denote r^0_{lt} and s^0_{lt} as the number of reparable and serviceable systems, respectively, that are in transit at the beginning of the planning horizon and will arrive at location l at the beginning of time period t. The terms r^{im}_{lt} and s^{im}_{lt} are defined similarly for modules of type i in state m.

It is important to note that the data required for this model are physical quantities and thus are easy to obtain or estimate based on current practices. For instances, existing models can be used to set inventory levels and transportation capacities can be based on current practice. The data elements ψ^n_{lt} present a unique challenge. Determining the number of time periods that a system will be used until it has reached its usable life is difficult to estimate. In the context of Air Force fighter aircraft, models have been developed that can be used to determine reasonable approximations for these values [13].

4.3.3 Basic Variables

In keeping with the notation from the initial conditions, we define $\mathbf{S^0_{lt}}$ and $\mathbf{R^0_{lt}}$ as the number of serviceable and reparable systems at location l at the beginning of time period t. For all modules of type i in state m, $\mathbf{S^{im}_{lt}}$ and $\mathbf{R^{im}_{lt}}$ are defined similarly.

Beyond the bookkeeping variables listed above, we must decide which modules to repair and how to assemble systems from the inventory of serviceable modules. The variable $\mathbf{x^{imm'}_{lt}}$ is the number of reparable modules of type i that enter the repair process in state m and complete repair in state m' beginning in period t at location l. This variable is defined only for $m' \geq m$. In the special case in which $m' = m$, $\mathbf{x^{imm}_{lt}}$ indicates the number of modules to move directly from the reparable inventory to the serviceable inventory. For system assembly, $\mathbf{y^{0n}_{lt}}$ is the number of systems to assemble in state n and test at location l beginning at time period t. If a location, such as the depot, does not perform system assembly and test, the $\mathbf{y^{0n}_{lt}}$ variables can be set to 0

or the γ_{lt} set equal to $T+1$. Similarly, $\mathbf{y_{lt}^{im}}$ is the number of modules of type i in condition m to pull from serviceable inventory and to install on systems at location l at time period t. To link module installation and system assembly, we define $\mathbf{w_{lt}^{imn}}$ as the number of modules of type i in state m to install on systems resulting in a system state of n at location l in period t. As a result of our assumption that the module with the minimum usable life determines the system usable life, the w variables are defined only for $n \leq m$.

To track the transport of systems and modules we define $\mathbf{r_{ll't}^{0}}$ as the number of reparable systems to transport from location l to location l' in time period t and $\mathbf{r_{ll't}^{im}}$ as the number of reparable modules of type i in state m to transport from location l to location l' in time period t. Both $\mathbf{s_{ll't}^{0n}}$ and $\mathbf{s_{ll't}^{im}}$ are defined similarly for serviceable systems and modules. With these choices of data and variables, we are prepared to formulate the Modular Maintenance and System Assembly model.

4.3.4 Objective Function and Individual Constraints

The model is intended to generate a maintenance and transportation plan that meets the demand for systems at each location while also maintaining an inventory of serviceable systems to account for random failures. However, having a stockpile of systems above this baseline at a location, while possibly mitigating risk, can incur costs due to storage, security, and other considerations. Our objective function will impose a cost (penalty) of α_{lt} for each serviceable system in place at location l at time period t below what is needed to meet external demand for systems. For this purpose, we introduce additional tracking variables $\mathbf{E_{lt}}$. The variables $\mathbf{E_{lt}}$ are defined as the demand minus the number of serviceable systems at location l in time period t. If this quantity is negative, (i.e., there are more serviceable systems than external demand) $\mathbf{E_{lt}}$ is defined to be 0. For a deficit of $\mathbf{E_{lt}}$ systems, the corresponding cost would be $\alpha_{lt} \cdot \mathbf{E_{lt}}$.

While this linear cost function captures the first order effect of too few serviceable systems, second order effects drive us to consider an increasing cost function. As an example, canceling one sortie due to engine maintenance has a small effect on the

long term performance of flight operations. However, canceling numerous sorites has a large effect, much greater than the combined effects of many single sortie cancelations. Accordingly, the steeper penalty for larger deviations is captured via a piecewise linear and convex cost function with variables $\mathbf{E_{ltj}}$, upper bounds $\mathcal{U}_{ltj}$ and coefficients α_{ltj}. The upper bound $\mathcal{U}_{ltj}$ limits the system shortfall that can be charged α_{ltj} and forces the formulation to move to the next higher linear piece, $j+1$. An example of the cost function is show in Figure 4-4. Note that in any optimal solution, $\mathbf{E_{ltj}} > 0$ implies $\mathbf{E_{lt(j-1)}} = \mathcal{U}_{lt(j-1)}$. Similarly, we use tracking variables $\widetilde{\mathbf{E}}_{\mathbf{ltj}}$, upper bounds $\widetilde{\mathcal{U}}_{ltj}$, and coefficients $\widetilde{\alpha}_{ltj}$ to penalize deviations from ξ_{lt}, the number of systems desired on hand to mitigate random failures or changes in demand.

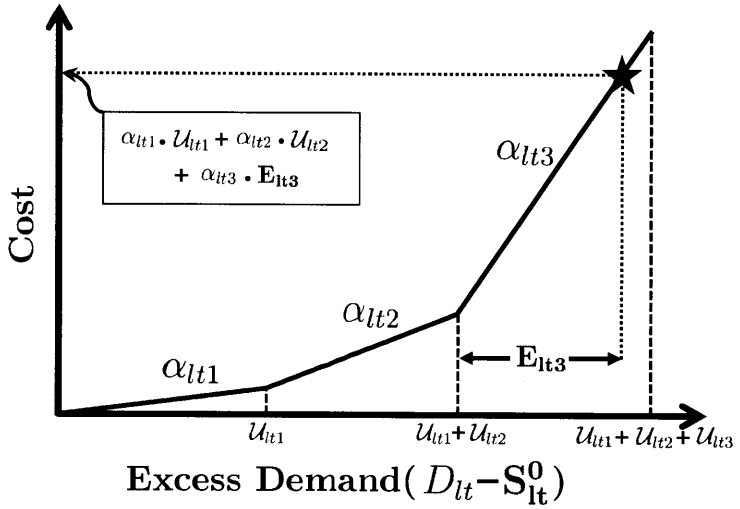

Figure 4-4: Objective function example

We will minimize the weighted sum of $\mathbf{E_{ltj}}$ and $\widetilde{\mathbf{E}}_{\mathbf{ltj}}$ over all locations, time periods, and deviation ranges $j \in J$. We envision that the weighting parameters α_{ltj} and $\widetilde{\alpha}_{ltj}$ would be increasing in j. Also we assume that $\alpha_{ltj} \geq \widetilde{\alpha}_{ltj}$ as a lack of serviceable systems to meet demand is much more detrimental than being below the serviceable system reserve level. Thus the objective function for our model is to:

$$\text{minimize} \sum_{l \in \mathcal{L}} \sum_{j \in J} \sum_{1 \leq t \leq T} \left(\alpha_{ltj} \cdot \mathbf{E}_{ltj} + \widetilde{\alpha}_{ltj} \cdot \widetilde{\mathbf{E}}_{\mathbf{ltj}} \right). \tag{4.1}$$

Constraints (4.2) ensure that the $\mathbf{E_{ltj}}$ variables cover any potential shortfall of serviceable systems. Similarly, any shortfall from the number of systems desired on hand to mitigate random failures, beyond that captured by the $\mathbf{E_{ltj}}$, is captured in the $\widetilde{\mathbf{E}}_{\mathbf{ltj}}$ variables in constraints (4.3). Constraints (4.4) and (4.5) limit the magnitude of each of these variables.

$$\mathbf{S^0_{lt}} + \sum_{j \in J} \mathbf{E_{ltj}} \geq D_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{4.2}$$

$$\mathbf{S^0_{lt}} + \sum_{j \in J} \left(\mathbf{E_{ltj}} + \widetilde{\mathbf{E}}_{\mathbf{ltj}} \right) \geq \xi_{lt} + D_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{4.3}$$

$$\mathbf{E_{ltj}} \leq \mathcal{U}_{ltj}, \qquad \forall\, l \in \mathcal{L},\ j \in J,\ 1 \leq t \leq T, \tag{4.4}$$

$$\widetilde{\mathbf{E}}_{\mathbf{ltj}}, \leq \widetilde{\mathcal{U}}_{ltj} \qquad \forall\, l \in \mathcal{L},\ j \in \widetilde{J},\ 1 \leq t \leq T. \tag{4.5}$$

The inventory of reparable systems at location l at the end of time period t, $\mathbf{R^0_{lt}}$, is the combination of eight terms:

1. reparable system inventory at the end of the previous time period,
2. the inflow of reparable systems that are either in use or assembly/test at the beginning of the planning horizon and will return for maintenance at time period t because they have reached the end of their usable life,
3. inflow of reparable systems that are in transit to location l at the beginning of the planning horizon and will arrive in period t,
4. reparable systems that are removed from the inventory and used for system assembly beginning in period t,
5. inflow of reparable systems that were assembled within the planning horizon, have reached the end of their usable life, and are returning for maintenance at location l in period t,
6. serviceable systems that were transported to location l from other locations and used at location l will enter maintenance in period t if the sum of the shipment

period, the transit time, and the expected number of usage periods given the system's arrival time is equal to t,

7. inflow of reparable systems into location l from other locations in period t, and

8. outflow of reparable systems from location l to other locations in period t.

The summation inside the parentheses of the fifth term removes the serviceable systems that were shipped to other locations after assembly and test at location l and thus will return for repair at the other location. Note that reparable systems shipped out of location l are removed from the inventory immediately however, reparable systems shipped into location l arrive after the transit time between locations l' and l, $\nu_{l'l}$. Putting these terms together yields the inventory equation for reparable systems.

$$\mathbf{R^0_{lt}} = \mathbf{R^0_{l(t-1)}} + A^0_{lt} + r^0_{lt} - \sum_{n\in\mathcal{N}} \mathbf{y^{0n}_{lt}} +$$
$$\sum_{n\in\mathcal{N}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \left(\mathbf{y^{0n}_{lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{0n}_{ll'(t'+\gamma_{lt'})}} \right) +$$
$$\sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}+\psi^n_{l(t'+\nu_{l'l})}=t}} \mathbf{s^{0n}_{l'lt'}} + \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{r^0_{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r^0_{ll't}}, \qquad \forall\, l\in\mathcal{L},\ 1\le t\le T. \tag{4.6}$$

The material balance equation for the inventory of serviceable systems is similar to the one for reparable system's inventory:

1. serviceable system inventory at the end of the previous time period,

2. outflow of serviceable systems that are either in use or assembly/test at the beginning of the planning horizon and must return for maintenance at time period t because they have reached the end of their usable life,

3. inflow of serviceable systems that were in assembly/test at the beginning of the horizon that come online in period t,

4. inflow of serviceable systems that are in transit at the beginning of the planning horizon and are scheduled to arrive at location l at period t,

5. systems for which assembly decisions were made during the planning horizon and will come online in period t,

6. serviceable systems that were assembled during the planning horizon and reach the end of their usable life in period t,

7. movement of serviceable systems into location l from other locations in period t, and

8. movement of serviceable systems out of location l into other locations in period t.

These terms combined yield the total inventory balance equation for serviceable systems.

$$\mathbf{S^0_{lt}} = \mathbf{S^0_{l(t-1)}} - A^0_{lt} + B^0_{lt} + s^0_{lt} + \sum_{n\in\mathcal{N}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}=t}} \mathbf{y^{0n}_{lt'}} - \sum_{n\in\mathcal{N}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \mathbf{y^{0n}_{lt'}} + \sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{s^{0n}_{l'lt'}} - \sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{0n}_{ll't}}, \qquad \forall\, l\in\mathcal{L}:\ l\neq 0,\ 1\le t\le T. \quad (4.7)$$

Maintenance limitations and leveling, enforced via upper and lower bounds on system's assembly and test requirements, are represented by constraints as shown below.

$$\sum_{n \in \mathcal{N}} \mathbf{y_{lt}^{On}} \leq U_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{4.8}$$

$$\sum_{n \in \mathcal{N}} \mathbf{y_{lt}^{On}} \geq Q_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{4.9}$$

$$\mathbf{y_{lt}^{On}} \geq Q_{lt}^{n}, \qquad \forall\, n \in \mathcal{N},\ l \in \mathcal{L},\ 1 \leq t \leq T. \tag{4.10}$$

Much like the expression for the inventory of reparable systems, the reparable module inventory material balance equation is quite complex. The first three terms are directly analogous to the first three terms in the reparable systems inventory material balance equation (inventory at the end of the previous time period, return of modules that were assembled into systems prior to the first time period that can enter maintenance beginning in period t, and reparable modules that are in transit in the first time period that arrive in period t). The remaining terms are:

4. modules that are entering the reparable pool in period t from systems that were assembled and used at location l,
5. reparable modules that enter maintenance in period t,
6. inflow of reparable modules from other locations in period t, and
7. outflow of reparable modules to other locations in period t.

From the group of modules used to assemble systems at location l, we subtract the modules from systems that were assembled at location l but were shipped to another location, which is the second term inside the parenthesis of term 4.

$$\mathbf{R^{im}_{lt}} = \mathbf{R^{im}_{l(t-1)}} + A^{im}_{lt} + r^{im}_{lt} +$$

$$\sum_{n\in\mathcal{N}} \left(\sum_{\substack{m'\in\mathcal{M}:\\ m'-n=m}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \mathbf{w^{im'n}_{lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{0n}_{ll'(t'+\gamma_{lt'})}} \right) - \qquad \forall\, i\in\mathcal{I},\ m\in\mathcal{M}:$$

$$m\neq 0,\ l\in\mathcal{L},$$

$$\sum_{\substack{m'\in\mathcal{M}:\\ m'\ge m}} \mathbf{x^{imm'}_{lt}} + \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{r^{im}_{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r^{im}_{ll't}}, \qquad 1\le t\le T. \tag{4.11}$$

Due to our conservative assumption that all modules contained in a transported serviceable system will enter the reparable inventory with zero life remaining, a slight addition is needed for the modular inventories for state zero for all module types. The last term in the constraint below directly captures this affect by adding modules to the zero state reparable pool for those systems that were transported from other locations into location l. In this term we capture the shipment time from l' to l as well as the expected usage time at l.

$$\mathbf{R^{i0}_{lt}} = \mathbf{R^{i0}_{l(t-1)}} + A^{i0}_{lt} + r^{i0}_{lt} +$$

$$\sum_{n\in\mathcal{N}} \left(\sum_{\substack{m'\in\mathcal{M}:\\ m'-n=0}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \mathbf{w^{im'n}_{lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{0n}_{ll'(t'+\gamma_{lt'})}} \right) -$$

$$\sum_{m'\in\mathcal{M}} \mathbf{x^{i0m'}_{lt}} + \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{r^{i0}_{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r^{i0}_{ll't}} +$$

$$\sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}+\psi^n_{l(t'+\nu_{l'l})}=t}} \mathbf{s^{0n}_{l'lt'}}, \qquad \forall\, i\in\mathcal{I},\ l\in\mathcal{L},\ 1\le t\le T. \tag{4.12}$$

The serviceable module inventory material balance equation is very similar to the serviceable system's inventory material balance equation. The first three terms

capture the inventory at the end of the previous time period, modules that are in repair at the beginning of the planning horizon and will enter the serviceable inventory in period t, and modules in transit at the beginning of the planning horizon that will arrive in period t. The remaining terms capture:

4. modules that are removed from the serviceable inventory for installation on a system in period t,
5. modules coming out of repair in period t,
6. inflow of serviceable modules from other locations in period t, and
7. outflow of serviceable modules to other locations in period t.

$$\begin{aligned}\mathbf{S}^{\mathbf{im}}_{\mathbf{lt}} = \mathbf{S}^{\mathbf{im}}_{\mathbf{l(t-1)}} + B^{im}_{lt} + s^{im}_{lt} - \mathbf{y}^{\mathbf{im}}_{\mathbf{lt}} + \\ \sum_{\substack{m' \in \mathcal{M}: \\ m' \leq m}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \tau^{im'm}_{lt'} = t}} \mathbf{x}^{\mathbf{im'm}}_{\mathbf{lt'}} + \\ \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \nu_{l'l} = t}} \mathbf{s}^{\mathbf{im}}_{\mathbf{l'lt'}} - \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}}, \qquad \forall\, i \in \mathcal{I},\ m \in \mathcal{M}, \\ l \in \mathcal{L},\ 1 \leq t \leq T. \end{aligned} \tag{4.13}$$

Limited maintenance resources constrain the total number of modules of a certain type that can be in maintenance in any period. The left-hand side of constraint (4.14) is the number of modules there were entered into repair at some period in the past and will still be in repair during time period t. The first term on the right is the maintenance capacity while the second term captures the modules that were in repair at the beginning of the planning horizon and will come out of repair during a future period.

$$\sum_{m \in \mathcal{M}} \sum_{\substack{m' \in \mathcal{M}: \\ m' \geq m}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \tau^{imm'}_{lt'} > t}} \mathbf{x}^{\mathbf{imm'}}_{\mathbf{lt'}} \leq M^{i}_{lt'} - \sum_{m \in \mathcal{M}} \sum_{t < t' \leq T} B^{im}_{lt'}, \qquad \forall\, l \in \mathcal{L},\ i \in \mathcal{I},\ 1 \leq t \leq T. \tag{4.14}$$

There are two main coupling constraints that bind the modular and system decisions together. The first of these constraints ensures that the number of modules assigned to systems in state n equals the number of systems in state n that are assembled. The second constraint ensures that the number of modules in state m assigned to system assembly equals the number of modules in that state that are pulled from the serviceable module inventory. The key to both of these constraints is the $\mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}}$ variables that directly link modules in a given state to systems in another state. Of note is the index of both summations only allows $m \geq n$ which implements our assumption that the module with the minimum usable life defines a system's usable life. In this formulation it is possible to assemble a system into state n even though all modules in the system have a usable life strictly higher than n. While assembling such a system might seem counterintuitive as doing so unnecessarily pulls systems from the serviceable inventory sooner than is required, such a decision might be optimal due to future maintenance constraints. As will be explored later, an artificial module can be included to maintain the convention that the usable life of a system is defined by the minimum of the usable lives of the modules that make up that system.

$$\sum_{\substack{m \in \mathcal{M}: \\ m \geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{0n}}, \qquad \forall\, l \in \mathcal{L},\ i \in \mathcal{I},\ n \in \mathcal{N},\ 1 \leq t \leq T, \tag{4.15}$$

$$\sum_{\substack{n \in \mathcal{N}: \\ m \geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{im}}, \qquad \forall\, l \in \mathcal{L},\ i \in \mathcal{I},\ m \in \mathcal{M},\ 1 \leq t \leq T. \tag{4.16}$$

To enforce our assumption that serviceable systems are transported only when coming out of assembly and test, we limit the number of serviceable systems that can be transported at a specific point in time (lefthand side) to be less than the number of systems coming out of assembly and test in that time period (righthand side).

$$\sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{s}_{\mathbf{ll't}}^{\mathbf{0n}} \leq \sum_{\substack{1 \leq t' \leq t: \\ t' + \gamma_{lt'} = t}} \mathbf{y}_{\mathbf{lt'}}^{\mathbf{0n}}, \qquad \forall\, n \in \mathcal{N},\ l \in \mathcal{L},\ 1 \leq t \leq T. \tag{4.17}$$

The flow of systems and modules, both reparable and serviceable, between loca-

tions is controlled by the following constraints. While these constraints are meant mostly for policy decisions (e.g., do not ship reparables between operating locations, but only between the operating locations and the depot) they can be used to enforce physical limitations for systems or physical modules (e.g., certain modules might only be transported only in cold storage or in a separate compartment due to security concerns).

$$\mathbf{r^0_{ll't}} \leq \mu^{0r}_{ll't}, \qquad \forall\, l,l' \in \mathcal{L}:\ l \neq l',\ 1 \leq t \leq T, \tag{4.18}$$

$$\sum_{n\in\mathcal{N}} \mathbf{s^{0n}_{ll't}} \leq \mu^{0s}_{ll't}, \qquad \forall\, l,l' \in \mathcal{L}:\ l \neq l',\ 1 \leq t \leq T, \tag{4.19}$$

$$\sum_{m\in\mathcal{M}} \mathbf{r^{im}_{ll't}} \leq \mu^{ir}_{ll't}, \qquad \forall\, l,l' \in \mathcal{L}:\ l \neq l',\ i \in \mathcal{I},\ 1 \leq t \leq T, \tag{4.20}$$

$$\sum_{m\in\mathcal{M}} \mathbf{s^{im}_{ll't}} \leq \mu^{is}_{ll't}, \qquad \forall\, l,l' \in \mathcal{L}:\ l \neq l',\ i \in \mathcal{I},\ 1 \leq t \leq T. \tag{4.21}$$

Aggregate capacity usage between locations for all systems and modules, both reparable and serviceable, is captured in the following constraint.

$$\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} \left(\mathbf{s^{im}_{ll't}} + \mathbf{r^{im}_{ll't}}\right) + \sum_{n\in\mathcal{N}} s^{\mathbf{0n}}_{\mathbf{ll't}} + \mathbf{r^0_{ll't}} \leq \mu_{ll't}, \qquad \forall\, l,l' \in \mathcal{L}:\ l \neq l',\ 1 \leq t \leq T. \tag{4.22}$$

As written, this constraint assumes that all modules and systems, whether reparable or serviceable, use the same amount of transportation capacity. While it is reasonable to assume serviceable and reparable modules of the same type use the same amount of transportation capacity and possibly that all modules use the same amount of capacity, it might not be reasonable to assume that modules and systems use the same amount of capacity. In the engine case we consider, a reparable system is solely a engine ID label thus it uses little to no transportation capacity. Serviceable engines, however, use a large amount of capacity. From a structural standpoint, adding a capacity coefficient to the serviceable systems' flow will introduce the only non-0,1 variable coefficient into the optimization.

Finally, all of the variables are constrained to be nonnegative integers. The integrality of these variables will be relaxed as part of the hierarchical decomposition developed in Chapter 5.

$$\mathbf{E}, \widetilde{\mathbf{E}}, \mathbf{R}, \mathbf{S}, \mathbf{y}, \mathbf{x}, \mathbf{w}, \mathbf{r}, \mathbf{s} \in \mathbb{Z}^{+}. \tag{4.23}$$

Combining all of these constraints with the objective function yields the full formulation which can be found in Appendix B. With this formulation in mind, we next consider how the tradeoffs discussed earlier are captured in the formulation.

4.4 Tradeoffs

As mentioned previously, at the planning level there are three main parameters that can be changed to affect the performance of the maintenance system as a whole: maintenance capacity, transportation resources (capacity and/or time), and inventory levels, both in the amount and the location at which those resources are located. An initial approach to this problem might separate the problem by location and solve the optimization problem assuming no flow of inventory occurs between locations. While such a situation is feasible with enough manpower and inventory, it cannot account for the load balancing that can occur between locations. For instance, when one location has a surge in demand, other locations can utilize their spare capacity to aid the overworked location. This not only helps the overloaded location meet the required demand, it also helps smooth the maintenance work being performed at the assisting locations. The formulation captures this tradeoff and allows planners to set an appropriate level of manpower and inventory at each location.

Fully sharing of maintenance resources among locations can be implemented in the extreme case by collocating all maintenance resources at a central depot (denoted as location 0 in the formulation). From constraints (4.13) and (4.16) we can see that in order to assemble a system, on hand serviceable inventory is needed. With a purely centralized maintenance capability, all serviceable modules must be transported to

and from each location. Accordingly, to return systems to a serviceable state in a timely manner will require frequent and reliable transportation between the depot and the operating locations as well as a steady resupply of serviceable modules at the operating locations. A more balanced approach to the problem would be to place enough maintenance resources at each location to meet the steady demand and have surge capacity available at the depot. While not removing the need for inventories at the operating locations or transportation between the depot and the operating locations, inputs to the formulation can be altered to select the appropriate mix of on-site and centralized repair based on the cost of transportation, inventory, and maintenance resources. In an Air Force context, the decisions where and how much inventory to hold at different locations will involve AFMC, ACC, and DLA.

The model also captures tactical tradeoffs at operating locations, which are of particular interest to an organization such as ACC. One such tactical tradeoff is the level of maintenance to perform on modules before placing them in the serviceable inventory. A reasonable policy would be to fully maintain each module to return it to the highest possible state. This would allow systems to function for the longest possible time and thus reduce the demand for test stand usage. It would also seem to keep a larger stockpile of serviceable systems available for use over the long term. However, from constraints (4.13), we see that a module returns to the serviceable inventory after $\tau_{lt}^{imm'}$ periods in maintenance. If m' is the highest module state, then the time in maintenance will increase thus delaying the reentry of the module into the serviceable pool.

Another issue that arises with such a policy is that the shop level maintenance constraints (4.14) limit the number of modules that can be in repair at one time. Many past models did not consider maintenance manpower limitations, essentially assuming a sufficiently large repair capacity. In such a situation, the inflow of modules into repair is not limited and the only impact on the serviceable module's inventory is the length of time needed to repair a module. Fully repairing each module thus does not impact the maintenance start times for subsequent modules. However, when maintenance manpower is limited, as it is in most cases, modules that are in repair affect the

inflow of modules into the repair process. Each module that is fully repaired spends longer in the maintenance shop than modules that are not fully repaired. This limits the inflow of other modules to the maintenance shop and thus further delays their return to the pool of serviceable modules. The combination of longer repair times and delayed entrance into maintenance can severely impact the serviceable module inventory thus necessitating a large supply of serviceable modules to reassemble reparable systems and return them to the serviceable systems inventory. These maintenance capacity and timing tradeoffs are not only of use at a tactical level but also at a planning and design level. If engine development teams at AFMC can reduce the time needed to perform specific maintenance actions through early investment in design and maintainability, these longer term impacts could be reduced significantly. This reiterates the importance of the model developed in Chapter 2 which allows near term investment and lifecycle maintenance costs to be considered jointly.

Another tradeoff, one that is frequently considered in operational settings, also pertains to the level of modular maintenance to perform. However, in contrast to the above scenario, we now consider a form of cannibalization, the practice of removing modules from a system, rather than from inventory, and using those modules to assemble another system. In our model, cannibalization can occur in the following way. When a system comes in for disassembly and repair, a module from that system will be added to the reparable module inventory. However, the decision to move that component directly into a system will be represented by $\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm}}$ and $\mathbf{y}_{\mathbf{lt}}^{\mathbf{im}}$ both being greater than or equal to one. Assuming $\tau_{lt}^{imm} = 0$, all of these actions will occur in the same time period which in a physical sense means the cannibalized part moves directly from the old system to the new system. From a tradeoff perspective, cannibalization is nothing more than deciding that the value of a module in its current state is greater than the future value it would have after some amount of maintenance. We again are trading between maintenance capacity, now and in the future, and both module and system inventory levels. Cannibalization is a deeply discussed topic in the Air Force with proponents for and detractors against cannibalization. By allowing planners to adjust the level of cannibalization allowed (through $\tau_{lt}^{imm} = 0$ and possibly upper

bounds on $\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm}}$) the impacts of cannibalization can be fully explored.

Reassembly of systems highlights another tactical tradeoff that can be explored with this model. The $\mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}}$ variables select which modules will be installed on systems resulting in a specific system state. Through constraints (4.7), and specifically $\psi_{lt\prime}^{n}$, a system will return for maintenance when the system has reached the end of its usable life n. When viewing the assembly problem independent of the system impacts, the optimal solution is to match the modules so as to have the m and n values as close as possible. This solution can be constructed by a simple greedy algorithm that assigns high state components together and low state components together. This, however, might be far from optimal due to the possible impacts of such a policy. When a system assembled under this policy returns for maintenance, all of the modules in the system will have very little usable life remaining thus necessitating higher levels of maintenance to return them to usable condition.

This effect is again evident through the $\tau_{lt}^{mm'}$ values and their impact on both the maintenance manpower constraints (4.14) as well as the serviceable inventory constraints (4.13). This policy might cause surges in the usage of maintenance resources in some periods while significantly under utilizing the same resources in other periods. Also, increasing the usable life of a component by a certain number of hours may require significantly longer time in maintenance depending on the starting state of the module. For instance, if a module has a reasonable number of hours of usable life remaining, a simple visual inspection might be enough to increase the usable life. However, if that same module has little to no usable life remaining, a complete rebuild or extensive non-destructive testing may be needed to clear the module for further usage.

It is possible to keep all modules at relatively high states of maintenance in the formulation by including an "artificial" module that has a large inventory, no maintenance limitations, and requires no lead-time to repair to a desired state. By installing one of these artificial modules on a system, the system and its corresponding modules will come in for repair at the desired time. We can also limit the time systems spend in operation without artificially reducing the maximum maintenance state for

modules by placing the appropriate upper bound on the usable life of the artificial module. All other system modules will continue to flow through the maintenance processes as normal.

4.5 Literature Review

Due to the large budgetary requirements associated with personnel, spare parts inventory, and transportation, there is a large literature that attempts to address portions of the problem. Beginning with the seminal work of Sherbrooke [42], the academic community has addressed the inventory policies for randomly failing, reparable parts. Sherbrooke's METRIC model assumes that module demand (failure) is compound Poisson and that repair can occur at the operating location with a certain probability. If the module cannot be repaired at the operating location, it must be shipped to a depot for repair. Due to the high cost of the modules, a one-for-one (i.e., $(s-1, s)$) inventory policy is employed. This implies that a requisition for a serviceable part is generated at the operating location whenever a reparable part is shipped to the depot for repair. The model does not, however, allow reparable or serviceable modules to be transferred between locations. In addition, the METRIC model implicitly assumes that there is an ample stock of repair parts (parts needed to repair a module) and that the repair times are independent for each module. From a practical standpoint, this assumption translates to a large, standing repair capability of inventory, manpower, and equipment. The objective of the METRIC model is to minimize inventory costs subject to expected backorders at the operating locations. The decision variables in the METRIC model are the appropriate inventory levels at the depot and each of the operating locations. To our knowledge, METRIC was the first multi-echelon model that was used by any organization for making procurement decisions.

Building upon the METRIC model, Muckstadt [34] considers an indentured parts structure in the MOD-METRIC model. Specifically, this model considers two levels, an engine and its modules. While the MOD-METRIC model maintains many of the assumptions in the original METRIC model, it makes a keen distinction that

engine and module backorders are not the same. In contrast to the METRIC, the MOD-METRIC model minimizes the expected backorders for engines subject to a budgetary constraint on engines and module inventories. As modules are required for engine assembly, the objective has an indirect effect upon module backorders. We note that the MOD-METRIC model was the first multi-echelon, multi-indenture model to be employed by the US Air Force for procuring inventories of modularly designed items.

Extending the MOD-METRIC model, Muckstadt [35] develops the Consolidated Support Model (CSM) to compare the two versus three level maintenance structure and the effects of centralized decision making with decentralized repair for a line replaceable unit (LRU) with a single module. Muckstadt uses a decomposition approach to solve the problem, first decomposing the LRU and module subproblems and secondly, decomposing the three-echelon problem into independent two-echelon subproblems. The solutions to each decomposition are appropriately combined to solve the initial problem. To solve the two-echelon module problem, Muckstadt [36] develops a computationally efficient method for determining optimal stock levels. By building investment tradeoff curves, the CSM algorithm can be extended to the case in which aircraft have multiple LRUs. These tradeoff curves allow planners to determine inventory investments, in aggregate, by LRU, and by module, to achieve specified backorder rates.

By using the Poisson failure and independent repair assumptions, one can view serviceable module shipments from the depot to operating locations as independent processes, and in particular, as Poisson processes. This is an obvious approximation as there is not an infinite repair capacity. Graves [24] extends the METRIC model by considering a repair facility with a finite capacity. Recognizing that such a model will not be tractable for general repair times, he approximated the module requisition process at the operating locations by a distribution that is fully defined by its first two moments, such as a negative binomial distribution. Computational tests indicate that this approximation more accurately models the capacitated situation than the Poisson approximation in the METRIC model.

Sherbrooke [43] extends the METRIC model by including both the multi-indenture and improved approximation results. Computational tests show that the so called VARI-METRIC model estimates engine backorders better than MOD-METRIC due to the improved approximation of the interaction of repair activity. In addition to the improved estimate of backorders, the approximation used in the VARI-METRIC model is simpler to calculate than the approximation proposed in Graves.

In more recent work, Díaz and Fu [17] and Díaz [16] have worked to improve the previous results for capacitated repair networks. In particular, they propose more complex approximations for the expectation and variance of the total number of modules in transit to and in repair at the depot. These approximation results are similar to the work of Graves but are based upon more recent queueing theory approximations. Poisson and general repair times are considered for a k server system for either a single class, or multiple classes, of reparable modules. Computational results indicate that the Díaz and Fu approximations improve upon the work of Graves.

There are many other extensions to the METRIC model, as well as other models for reparable parts, that are beyond the scope of this review. We refer the reader to the excellent compilations of Muckstadt [38] and Sherbrooke [44] for further discussions about reparable inventory models.

While the METRIC model and its extensions are useful in setting initial inventory levels, there are three main issues that must be addressed for preventative maintenance applications. To aid in the analysis of the MOD-METRIC and VARI-METRIC models, researchers have assumed that only one module will require base level repair when an engine comes out of service. While such an assumption might be valid when considering failed modules, it becomes less valid when we consider preventative maintenance in which we might repair multiple modules simultaneously before they fail. In addition, the models discussed above assume that an $(s-1, s)$ inventory policy is appropriate for reparable modules. While such a policy might be appropriate with well developed infrastructure, in austere conditions in which transportation resources are limited, such a policy may in fact be infeasible. Finally, the models above aid in

setting initial inventory levels. They do not, however, aid in the maintenance decision making process. In particular, if multiple modules, or modules of the same type in different states, use the same maintenance resources, maintenance decisions must be made in addition to the inventory levels.

While the METRIC model focuses on the inventory procurement problem, another stream of academic research that focuses upon the distribution of reparable inventory, based on the work of Miller [33], is relevant to the engine problem. Miller considers the single module repair problem with multiple bases and a single repair depot. He assumes that modules fail according to base dependent Poisson processes and are repaired at the depot according to another, independent, Poisson process. In addition, he considers deterministic transportation times between the depot and the bases. The objective in the model is to minimize the total number of backorders at the bases. An implicit assumption made in the model is that the base Poisson processes are independent of past inventory distribution decisions. From an operational context, this implies that a backorder does not prevent operations but rather is a lack of reserve inventory at the base.

Recognizing that the state space of the model is large, Miller proposes the "Transportation Time Look Ahead Policy" which assigns modules coming out of repair at the depot via a myopic policy. In particular, the expected number of backorders at each base is calculated for the arrival period of the newly repaired model (i.e., the current period plus the base dependent transit time). The newly repaired module is shipped to the base for which the expected number of backorders has the largest marginal decrease. While this is a heuristic solution to the original problem, Miller shows that the "Transportation Time Look Ahead Policy" is optimal for a slightly modified model in which there exists sufficient reparable inventory and maintenance capacity. Specifically, he assumes that any demand at a base coincides with an item becoming available at the depot.

Miller and Modarres-Yazdi [32] build upon the original analysis by showing that a modified version of the "Transportation Time Look Ahead Policy" is optimal in the limit as the number of bases goes to infinity. The expanded analysis again uses

the assumption that the base failure rates are unaffected by inventory distribution decisions. Using results from queueing theory, they establish a long term average for the number of modules en route to the depot. This characterization of the average en route inventory allows them to show the optimality of the "Transportation Time Look Ahead Policy". As a byproduct of the analysis, they also show that the initial inventory assignments under the METRIC model also yield an optimal policy when the number of bases is large.

Three main practical issues arise with these two distribution models. While they consider transportation time for serviceable inventory between the depot and the bases, they do not consider the transit time from the bases to the depot. In addition, the assumption that the removal rate at a base is independent of past decisions is valid only if every base receives enough inventory assignment to continue full operations. If, for instance, a base receives too little serviceable inventory, operations might be curtailed as there will not be sufficient inventory to fully maintain aircraft. This leads to the final point in that both models consider every backorder to have the same consequence. In an operational setting, a backorder that grounds an aircraft is significantly more detrimental than a backorder that reduces the back shop inventory by one.

To address the concerns listed above, among others, the Air Force and RAND have published numerous reports aimed at addressing implementation in the military context. From a policy point of view, Air Force Instruction 21-129 (1998) gives some of the most direct guidance for maintenance of reparable systems. In particular, it lays out a two level maintenance (2LM) strategy whereby repairs will occur either on aircraft or at a centralized depot location. This is in contrast to the three level maintenance strategy that had been followed in the past. Under the 2LM policy, intermediate maintenance work will no longer be conducted at a base. Rather, reparables will be sent to the depot for repair. The authors of AFI 21-129 recognize the importance of an efficient transportation network to support this maintenance concept. It sets a goal of shipping a reparable to the depot within 48 hours of removal from the aircraft. In the context of engine maintenance, base repairs will occur

at the engine level. Any module that requires further repair would be sent to the depot for maintenance. While AFI 21-129 stresses the importance of the transportation network, it does not give policy makers methods to determine the effectiveness of the transportation network and its impacts on the execution of the 2LM concept.

This point is further driven home by Amouzegar et al. [7] who consider alternatives for intermediate jet engine maintenance. They specifically identify inter- and intratheater transportation as key to each of their alternatives for engine maintenance. In reference to their recommended maintenance policy for a large class of engines, they specifically state that substantial dedicated intratheater transportation is required and that this policy is sensitive to transportation times. They draw these conclusions from a simulation model based on a limited amount of historical data.

More recently, a number of RAND reports have addressed the repair issue with the aging fleet of aircraft, the increased cost/decreased availability of maintenance manpower, and the change in engagement strategy resulting from, specifically, the Air and Space Expeditionary Force (AEF) concept. Geller et al. [22] consider a case in which the operating location is unknown but due to lead times for setting up on-site maintenance activities (i.e., construction of an engine test cell requires approximately 30 days), centralized intermediate repair facilities (CIRF) are examined. They used a simulation model to develop recommendations for a variant of the F100 engine and for an electronic counter measures (ECM) pod. In both cases the CIRF concept proves valuable. Engine removal rates were relatively low. Thus, the simulation results indicated robustness against transportation times but were strongly dependant upon the number of initial spares. With higher removal rates, the ECM pod repair network was more sensitive to transit times and was also sensitive to the number of initial spares. Short of numerous runs, the simulation does not provide decision makers with the ability to understand changes to the repair concept and to examine changes in transportation and spares policies.

Other works have used analytical methods to address the maintenance problem. Keating et al. [26] use a network queuing model to determine the effect of depot-level capacity on the long-term affordability of a large transport aircraft. They specifically

do not consider the investments needed to attain the required capacity nor do they consider the tradeoff between spare parts and maintenance equipment. Loredo et al. [29] use queueing theory to analyze the periodic inspection and repair of structural elements of a fleet of aging aircraft. These inspection/repair actions are labor intensive, requiring anywhere between 2,000-50,000 man hours to complete. They develop bounds on the aircraft output rate using multi-server queuing theory.

Recently, focus has shifted to decision support tools that guide policy makers when making investment decisions. Using integer programming, McGarvey et al. [31] consider the assignment of forward operating locations' maintenance requirements to centralized repair facilities (CRF) for which the fixed (buildings, equipment, etc.) and variable (repair teams) capacities must be determined. They used linear regression to estimate the manpower requirements at the CRFs and forward operating locations. Tripp et al. [46] use a mixed integer formulation to make tradeoffs among personnel, transportation, and facility costs. They do not specify the formulation that is used but show that the cost of the optimal solution, which only places one CRF, is dominated by labor costs. Risk mitigation might necessitate multiple CRFs, but this will have cost consequences. In this case, the minimum cost solution that chooses at least two CRFs (it in-fact selects exactly two CRFs) has a cost that is less than 1% higher than the original solution.

Neither McGarvey et al. [31] nor Tripp et al. [46] discuss the computational effort required to solve their formulations. Given the underlying covering problems both must solve, we would not expect either to solve quickly except in very small instances. Both models focus on a long-term planning problem that will be solved infrequently. They do not, however, address operational or tactical problems that must be solved on a recurring basis.

While much of the above work focuses on long-term, steady-state models, Hillestad and Carrillo [25] and Muckstadt [37] examine the adequacy of of such models under dynamic scenarios in which flying operations change quickly and/or drastically. Hillestad and Carrillo describe several time dependent measures for nonstationary inventory systems. While some of the measures are drawn from classical inventory

literature, others such as the average number of systems not mission capable for supply (both with and without cannibalization) are specific to Air Force applications. They then show how to characterize the distribution of the number of serviceable systems under a nonhomogeneous Poisson process and give closed form solutions for specific examples of interest.

Muckstadt [37] analyzes a two-echelon inventory model for recoverable items. Using results from queueing theory, he approximates the distribution of the number of units in resupply, at the depot and the bases, when the failures at each base are independent, nonstationary Poisson processes under a continuous time model. He then develops a discrete time model that exactly characterizes the distribution of the number of units in resupply at each location. Finally, Muckstadt compares the results from these two models on sample data from F-15 avionics modules. The data represents an initial surge in flying activity that tapers off, returning to the stationary level after 30 days. These results indicate that the approximation closely matches the exact distribution across a range of failure rates, base repair probabilities, and depot stock levels. Computation times for the approximation were less than a second while calculating the exact distribution took 500 to 2000 times longer. Similar to previous work, a key assumption for both of the nonstationary models is an ample repair capability which results in independence between the repair and failure processes.

4.6 Summary

In this chapter we described the ingredients of a comprehensive module repair and system assembly problem. With the intent of tying planning and operations together, as well as making tradeoffs among maintenance capacity, transportation resources, and on hand inventory, we formulated a large integer program that seeks to fulfill demand for systems by repairing modules, assembling systems, and transshipping many types of inventory. Due to the complexity of the formulation, it is unlikely that a realistic sized instance could be solved directly with commercial integer programming software. To address this issue, and to provide planning and operational schedules

tools, in Chapter 5 we develop a decomposition scheme and specialized algorithms.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Decomposition and Solution Methodologies

When considering the full Modular Maintenance and System Assembly problem, we are motivated to consider a hierarchical solution methodology for four main reasons. First, the full model is computationally intractable across a long planning horizon with a small enough time increment as to adequately capture the intricacies of the entire assembly and repair process. Second, inherent randomness will occur in the actual problem. Incorporating random failures in the full model would exacerbate the computational issues. Based on discussion in Section 4.1, the different organizations involved in decision making is a third reason for the decomposition. Finally, there are different planning horizons that are of interest when considering module repair and system assembly. In particular, the effects of system assembly decisions arise much sooner than those of module repair.

We thus consider three types of models, (i) the full *planning model* that is solved infrequently over a long horizon with a relatively course time increment and continuous variables, (ii) an integral *system assembly model* solved frequently and over a shorter time horizon, and (iii) integral *module repair models* (one for each module type) that are also solved frequently and over a planning horizon that is longer than the horizon for the system model but shorter than the full planning model's horizon.

The idea of hierarchical decompositions has been exploited heavily in a number of

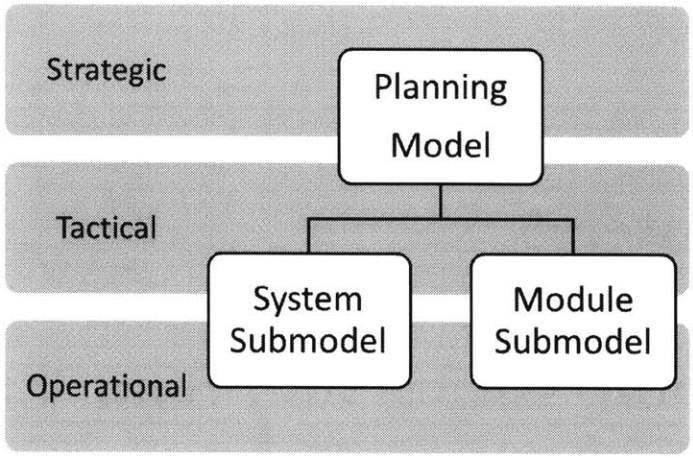

Figure 5-1: Hierarchial Decomposition

application environments. Based on the inventory and transportation decisions in our model, production planning problems are the most relevant regime comparisons for which a rich academic literature exists. Of particular interest are works of Graves [23] and Bitran et al. [10] who study hierarchical production planning problems. Graves decomposes the problem as an aggregate planning subproblem and a disaggregate subproblem linked by Lagrange multipliers. Bitran et al. use linear programs and knapsack problems to solve an aggregate problem and then disaggregate the solution. A more complete review of hierarchical production planning can be found in Bitran and Tirupati [11].

Figure 5-3 shows the decision flow beginning with the different models we develop. While the system and module submodels can be solved in parallel, another solution approach is to solve the system submodel first and then use the results from that model as input data for the model repair submodels. Later, we will develop efficient solution methodologies for the system assembly problem. Thus, this loss of parallelization will have only a small effect on the solution times.

A main reason that such a sequential approach is useful is the disparate time intervals over which the impacts for different decisions take to be realized. For instance, for many modules, the remaining usable life can be many months or even a few years. Repair of modules requires weeks or months to complete. System assembly and test

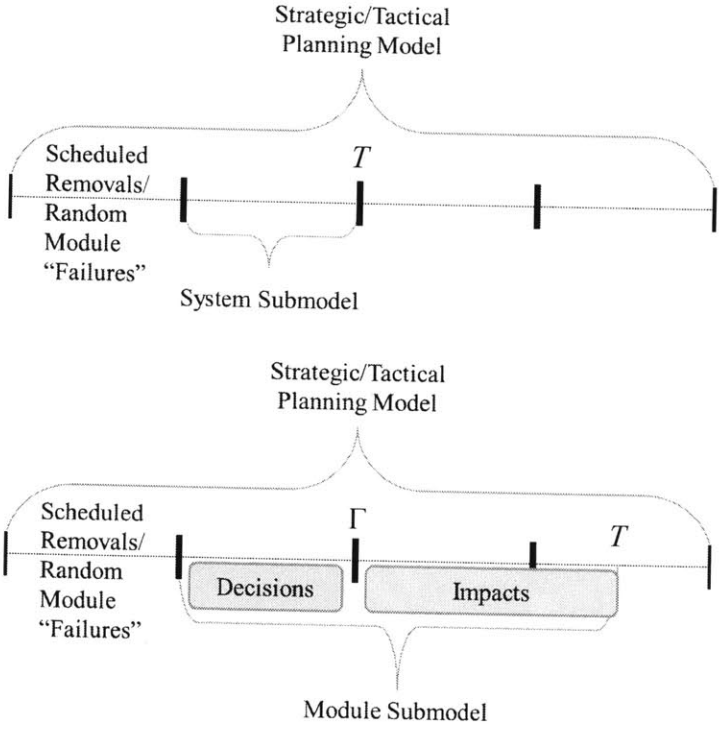

Figure 5-2: Planning model, system assembly, and module maintenance timelines.

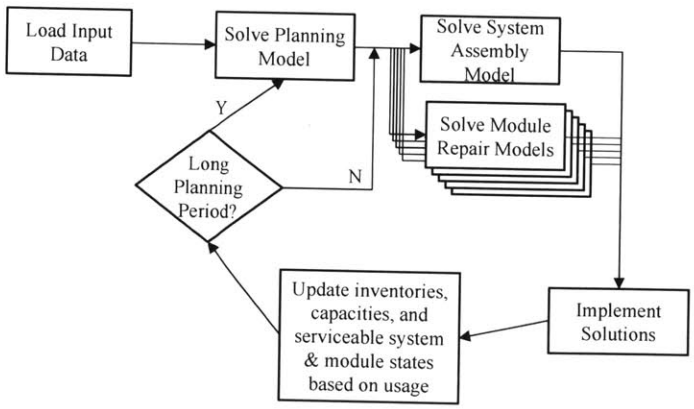

Figure 5-3: Execution environment for hierarchical models.

can be completed in days or a week. These different time increments can be used to model the repair process at the appropriate level for each phase of operation while still allowing for long term impacts to be considered. Long term impacts of near term decisions in shorter horizon models will be captured through target inventory levels, both system and modules, that will be passed to the shorter term planning models. The inventory levels of specific interest are the serviceable inventories, $\mathbf{S}$, for both the system and modules. In addition, we will obtain nominal system and module return rates $\mathbf{y_{lt}^{0n}}$ and $\mathbf{w_{lt}^{imn}}$ respectively. These return rates represent the inflow of reparable systems and modules based on the maintenance decisions made in the planning model.

The system submodel will use the serviceable system inventories by state, $\overline{S_{lt}^{0n}}$ as a target for the end of the shorter planning horizon. These inventory targets can be calculated from the planning models solution, namely the $\mathbf{S_{lt}^{0}}$ (serviceable system inventories), $\mathbf{y_{lt}^{0n}}$ (system assembles in state n), $\mathbf{s_{ll't}^{0n}}$ (serviceable system transshipment) variables and the usage rates ψ_{lt}^{n}. Figure 5-4 shows the transformation of these inputs from the planning model into the serviceable system targets for the system assembly submodel. The objective of our formulation will be to match the serviceable system targets and thus we penalize deviations from those targets. Small differences from the targets have little impact on the long term functioning of the repair and assembly process. Differing by a large amount, however, has drastic consequences. Deviations above and below the target value are both counterproductive. Accordingly, we define a convex cost function $\mathfrak{G}_{lt}^{0n}(\cdot)$, with $\mathfrak{G}_{lt}^{0n}(0) = 0$, that imposes a cost for deviating from the target value. As the purpose of $\mathfrak{G}_{lt}^{0n}(\cdot)$ is to incentivize matching the serviceable system inventory targets, we assume that $\mathfrak{G}_{lt}^{0n}(\cdot)$ has a unique minimum at zero. While the uniqueness property is not central to the general problem, it will be needed for some of the algorithms developed later in this section.

While the specific form of $\mathfrak{G}(\cdot)$ depends upon the application, we imagine that a reasonable condition is that $\mathfrak{G}_{lt}^{0n}(\cdot)$ is non-decreasing in the system state n for negative values and non-increasing in n for positive values. The intuition being that having too few high remaining life systems is more detrimental than having too many. In

the same vein, having too many low remaining life systems is more detrimental than having too few. Finally, in similar context to a discount factor, we would imagine that $\mathfrak{G}_{lt}^{0n}(\cdot)$ is non-increasing in t, that is, missing targets far in the future is less costly than missing near-term targets. The reasoning behind such a characterization comes from the fact that we will solve the system submodel on every day or every few days. Thus, we can address long term issues in future time periods while near-term decisions will be implemented and the consequences felt.

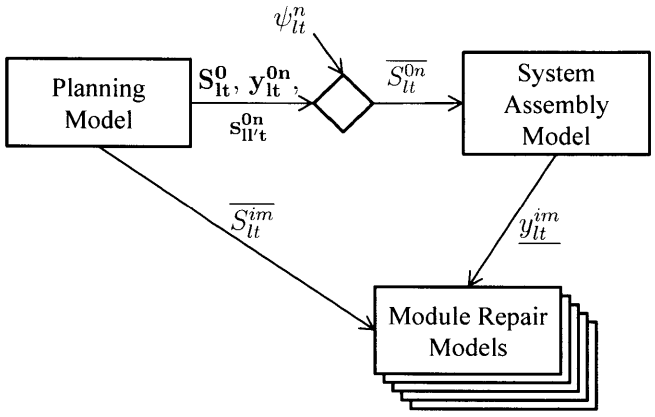

Figure 5-4: Model Interactions

Due to the longer time periods required for module repair, the planning horizon for the module submodel can be divided into two, usually uneven, parts. During the first part of the planning horizon, maintenance decisions are made and modules entered into the repair process. This short term decision horizon will match closely with the planning horizon for the system submodel. The results of the decisions made during the first part of the planning horizon are captured in the second portion of the planning horizon when modules are returned to the serviceable module inventory after being repaired. A constant over all of the time periods will be the maintenance capacity constraints. Modular repair decisions made during the planning model will be incorporated into the module submodels for time periods after the initial decision making phase. Similar to the system submodel, we define $\overline{S_{lt}^{im}}$ and $\mathfrak{G}_{lt}^{im}(\cdot)$ for each serviceable module inventory of type i in state m. The same characterization and

intuition discussed previously apply to the penalty function. If the system assembly and module repair models are implemented in sequence (i.e., the system assembly model is solved and then the module repair models are solved), serviceable module usage from the system assembly model will be passed to the module repair models in the data element $\underline{y_{lt}^{im}}$ which is an exact transformation of the system assembly decision variables $\mathbf{y_{lt}^{im}}$.

5.1 Planning Model Formulation

The planning model formulation is exactly the same as the full formulation described earlier except that the integrality constraints (4.23) are replaced by nonnegativity constraints:

$$\mathbf{E}, \widetilde{\mathbf{E}}, \mathbf{R}, \mathbf{S}, \mathbf{y}, \mathbf{x}, \mathbf{w}, \mathbf{r}, \mathbf{s} \in \mathbb{R}^{+}. \tag{5.1}$$

While the resulting linear program can be quite large, the constraints have sufficient structure (i.e., there is an underlying network structure and the constraint matrix is relatively sparse) so that the problem can be solved efficiently by a commercial linear programming package. In fact, we were able to solve realistic sized instances with fifteen modules, ten locations, ten module maintenance states, and a planning horizon of one year (one time period representing a week), were solved in 5-8 minutes using CPLEX 11.21 on a computer with a 6-core, 2.8 GHz CPU, and 30 GB of RAM.

From a planning standing, the outputs of most interest from this model are $\mathbf{E}$ and $\widetilde{E}$ variables which indicate the unmet demand and unmet safety stock at each location and time period. In addition, planners may be interested in the inventory constraints (4.7) and (4.13), maintenance capacity constraints (4.8) and (4.14), and the transportation capacity constraints (4.17) through (4.22). In particular, the dual variables for these constraints can guide planners in deciding where to allocate additional resources. In addition, as indicated in Figure 5-4, decision variables from the planning model will be passed to the system assembly and module repair submodels

in the form of serviceable inventory targets.

5.2 System Assembly Submodel

The system assembly submodel enables operational implementation of the planning model solution. This model seeks to match the serviceable system inventories from the planning model over a short time horizon. To meet this goal, the system assembly model considers the inventories of reparable systems and serviceable modules and assembles systems in varying states at different locations. It uses serviceable modules, which have been repaired in previous time periods, to assemble systems while respecting the assembly/test capacity constraints, both upper and lower bounds. As a reminder, the upper bounds represent assembly/test capacity constraints while the lower bounds represent the desire for maintenance personnel to remain proficient. Repair decisions for modules are made as part of the module submodels discussed in Section 5.3.

While a time period in the planning model might represent a week or longer, the operational decisions in the system assembly model drive us to consider a more granular set of decisions. In this case, a time period would represent one, or at most, a few days with a planning horizon of a few weeks. Such a relatively short planning horizon might seem quite myopic considering the long-term impacts of near term assembly decisions. The long-term impacts are exactly the reason for using the serviceable system inventory targets from the planning model, $\overline{S_{lt}^{0n}}$. These targets directly incorporate future maintenance repercussions of assembly decisions while also working to meet the near-term operational demand for systems.

Using the results from the planning model we can now consider the system assembly problem. Due to the frequency with which the system assembly submodel is solved, we will consider the serviceable system inventory only at the end of a relatively short planning horizon. Accordingly, the objective function of this model will include only $\mathfrak{G}_{lT}^{0n}(\cdot)$, $\overline{S_{lT}^{0n}}$ (input from the planning model), and the variable $\mathbf{S_{lT}^{0n}}$ for a planning horizon of T periods. Similarly, the only serviceable system inventory variables that

will be considered are $\mathbf{S_{lT}^{0n}}$, the number of systems assembled in state n in the system assembly submodel. This results in a formulation similar to the planning model with specific changes in constraints (5.3) and (5.7).

The planning model does not directly deal with stochastic failures. Rather, an excess of serviceable systems is desirable to mitigate potential failures. Due to the extremely low probability of a module failing that causes a system failure, we assume there are no system failures. The randomness we consider can be characterized as follows: when a system is pulled from service due to zero usable life remaining, modules in that system might be in a different state than originally forecast due to stochastic degredation. In this sense, a failure does not imply that a module is unusable but rather it may have degraded further than would be expected. For instance, metallic parts under vibration may begin to crack. Depending upon the component, the size of the cracks observed determine the remaining usable life of the module containing that component. These random "failures" will be considered indirectly in the system submodel as the number of reparable and serviceable systems and the serviceable module inventory may be different than originally planned. By solving the system submodel on a recurring basis, we can mitigate these random occurrences by taking appropriate actions to return the system inventories to those forecasted by the planning model.

The remainder of the variables are defined the same as in the planning model formulation:

- $\mathbf{R_{lt}^{0}}$ the reparable system inventory,
- $\mathbf{y_{lt}^{0n}}$ the number of systems to assemble in state n at location l beginning in time period t,
- $\mathbf{y_{lt}^{im}}$ the number of serviceable modules of type i in state m to use in assembling systems at location l beginning in time period t,
- $\mathbf{w_{lt}^{imn}}$ the number of serviceable modules of type i in state m to assemble into systems yielding a system state of n at location l beginning in time period t,

- $\mathbf{r^0_{ll't}}$ the number of reparable systems to ship from location l to location l' in period t, and
- $\mathbf{s^{0n}_{ll't}}$ the number of serviceable systems in state n to ship from location l to location l' in period t.

Minimize weighted difference from planning model targets

$$\text{minimize} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{N}} \mathfrak{G}^{0n}_{lT} \left(\overline{S^{0n}_{lT}} - \mathbf{S^{0n}_{lT}} \right) \tag{SSmP}$$

subject to

Reparable systems inventory: inventory from previous period, plus systems coming out of service and in transport, minus systems assembled, plus net inflow of reparable systems

$$\mathbf{R^0_{lt}} = \mathbf{R^0_{l(t-1)}} + A^0_{lt} + r^0_{lt} - \sum_{n \in \mathcal{N}} \mathbf{y^{0n}_{lt}} + \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \nu_{l'l} = t}} \mathbf{r^0_{l'lt'}} - \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{r^0_{ll't}} \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.2}$$

Serviceable systems inventory: initial inventory, inflow from assembly/test and other locations, assemblies during the planning period minus systems shipped to other locations, plus inflow of systems from other locations

$$\mathbf{S_{lT}^{0n}} = \sum_{\substack{n' \in \mathcal{N}: n' > n \\ \psi_{l0}^{n'-n} \leq T \\ \psi_{l0}^{n'-n+1} > T}} S_{l0}^{0n'} + \sum_{\substack{n' \in \mathcal{N}: \\ n' > n}} \sum_{\substack{1 \leq t \leq T: \\ \psi_{lt}^{n'-n} \leq T-t \\ \psi_{lt}^{n'-n+1} > T-t}} \left(B_{lt}^{0n'} + s_{lt}^{0n'} \right) +$$

$$\sum_{\substack{n' \in \mathcal{N}: \\ n' > n}} \sum_{\substack{1 \leq t \leq T: \\ \psi_{l(t+\gamma_{lt})}^{n'-n} \leq T-t-\gamma_{lt} \\ \psi_{l(t+\gamma_{lt})}^{n'-n+1} > T-t-\gamma_{lt}}} \left(\mathbf{y_{lt}^{0n'}} - \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{s_{ll'(t+\gamma_{lt})}^{0n'}} \right) + \qquad \forall\, n \in \mathcal{N},\ l \in \mathcal{L}:\ l \neq 0,$$

$$\sum_{\substack{n' \in \mathcal{N}: \\ n' > n}} \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \sum_{\substack{1 \leq t \leq T: \\ \psi_{l(t+\nu_{l'l})}^{n'-n} \leq T-t-\nu_{l'l} \\ \psi_{l(t+\nu_{l'l})}^{n'-n+1} > T-t-\nu_{l'l}}} \mathbf{s_{l'lt}^{0n'}}, \qquad 1 \leq t \leq T, \tag{5.3}$$

Test stand capacity constraint

$$\sum_{n \in \mathcal{N}} \mathbf{y_{lt}^{0n}} \leq U_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.4}$$

Maintenance proficiency requirement

$$\sum_{n \in \mathcal{N}} \mathbf{y_{lt}^{0n}} \geq Q_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.5}$$

State specific maintenance proficiency requirement

$$\mathbf{y_{lt}^{0n}} \geq Q_{lt}^{n}, \qquad \forall\, n \in \mathcal{N}, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.6}$$

Serviceable module inventory: initial inventory plus net inflow from repair and other locations

$$\mathbf{S_{lt}^{im}} = \mathbf{S_{l(t-1)}^{im}} + B_{lt}^{im} + s_{lt}^{im} - \mathbf{y_{lt}^{im}}, \qquad \forall\, i \in \mathcal{I},\ m \in \mathcal{M},\ l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.7}$$

System build constraints

$$\sum_{\substack{m\in\mathcal{M}:\\ m\geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{0n}}, \qquad \forall\, l \in \mathcal{L},\ i \in \mathcal{I},\ n \in \mathcal{N},\ 1 \leq t \leq T, \tag{5.8}$$

Module balance constraints

$$\sum_{\substack{n\in\mathcal{N}:\\ m\geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{im}}, \qquad \forall\, l \in \mathcal{L},\ i \in \mathcal{I},\ m \in \mathcal{M},\ 1 \leq t \leq T, \tag{5.9}$$

Serviceable systems shipped only coming out of assembly and test

$$\sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s}_{\mathbf{ll't}}^{\mathbf{0n}} \leq \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}=t}} \mathbf{y}_{\mathbf{lt'}}^{\mathbf{0n}}, \qquad \forall\, n \in \mathcal{N},\ l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.10}$$

Reparable systems flow

$$\mathbf{r}_{\mathbf{ll't}}^{\mathbf{0}} \leq \mu_{ll't}^{0r}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l',\ 1 \leq t \leq T, \tag{5.11}$$

Serviceable systems flow

$$\sum_{n\in\mathcal{N}} \mathbf{s}_{\mathbf{ll't}}^{\mathbf{0n}} \leq \mu_{ll't}^{0s}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l',\ 1 \leq t \leq T, \tag{5.12}$$

Nonnegativity and integrality of decision variables

$$\mathbf{R}, \mathbf{S}, \mathbf{y}, \mathbf{x}, \mathbf{w}, \mathbf{r}, \mathbf{s} \in \mathbb{Z}^{+}. \tag{5.13}$$

Even using the decomposition approach, the system submodel could be difficult to solve due to the number of modules, repair states, and locations. If the system submodel is to be used in planning, as well as execution, efficient solution algorithms are needed to allow policy makers to make tradeoffs and evaluate the impacts of their changes. In the following sections we develop three efficient algorithms individually through a special case when all modules are in the same state, by relaxing the system

assembly capacity constraints (5.4) through (5.6), and finally, by assuming the cost functions $\mathfrak{G}(\cdot)$ have a specific structure, namely a "V" shape. If the transit times are relatively long, a simplifying assumption that could be made, due to the relatively short time horizon over which the system assembly submodel, is to remove the inventory flow decision between locations. This allows us to solve the problem for each location separately. If, on the other hand, transit times are negligible, separate locations can be combined into a single location. In either case, we are left with a problem without transhipment considerations. The second and third algorithms make use of this fact by focusing on solving the system submodel for an individual location.

5.2.1 Solving the System Submodel: A Network Flow Methodology for Policy Tradeoffs

The system assembly submodel is hard to solve due to both the assembly capacity constraints (upper (5.4) and lower (5.5), (5.6) bounds) and the build (5.8) and balance (5.9) constraints. However, by making a simple assumption concerning the build and balance constraints, the model becomes efficiently solveable and can be used to make policy tradeoffs between system assembly capacity, transportation capacity, and system and module inventory. In particular, by assuming that every serviceable module is at the same state m, we can transform the system assembly problem into a network flow problem that can be solve efficiently. Next we will describe the nodes, sources/sinks, arcs, arc flow bounds, and the arc flow costs, in the network formulation. The network flow problem will contain seven sets of nodes with each location having separate nodes except as indicated.

1. **Initial Serviceable System Inventories:** The nodes in this set represent the initial inventory of serviceable systems in each state. The network contains one such node for each location and system state.

2. **Assembly Inventory:** These nodes represent the inventory of reparable systems and serviceable modules that can be used to assemble a serviceable system in each time period.

3. **System Assembly:** These nodes represent the decision to assemble systems using reparable systems and serviceable modules in a given time period.

4. **State Specific System Assembly:** These nodes represent the decision to assemble systems into a particular state in each time period.

5. **Final Serviceable System Inventories:** Nodes in this set represent the inventory of serviceable systems in each state at the end of the planning horizon.

6. **Target Serviceable System Inventories:** As our intent is to match the serviceable systems inventory targets provided by the planning model, these nodes allow us to incur costs for deviating from those targets.

7. **Sink:** This node serves as a sink for all flow in the network.

Of the seven sets of nodes, only two sets (1 and 2) serve as sources, and only the final one serves as a sink, for which the outflow will be equal to the sum of all sources. The initial serviceable system inventory nodes will have an inflow equal to the initial serviceable systems inventories at location l and state n, denoted as S_{l0}^{0n}. Assembly inventory nodes will have of inflow equal to the number of reparable systems and serviceable modules available by location and time period. By assumption, all of the modules are in the same state. Thus, there is only one inventory for each module type $i \in \mathcal{I}$. Consider the first time period of the planning horizon. In that period, the maximum number of systems that can be assembled at a location is the minimum of the reparable systems inventory and the number of modules of each type. In subsequent periods, the number of additional systems that can be assembled at a location, disregarding inflow from other locations, is the incremental increase in that minimum from the previous period. Recall that module repair decisions are made in the planning model and module submodels and thus are considered as data in the system submodel. Using the notation described above, we can define the assembly inventory source at a given location l for period 1,

$$b_{l1} = \min\left\{R_{l0}^{0} + A_{l1}^{0} + r_{l1}^{0}, \min_{i\in\mathcal{I}}\left\{S_{l0}^{im} + B_{l1}^{im} + s_{l1}^{im}\right\}\right\}, \tag{5.14}$$

and for subsequent periods t,

$$b_{lt} = \min\left\{R^0_{l0} + \sum_{1\le t'\le t}\left(A^0_{lt'} + r^0_{lt'}\right), \min_{i\in\mathcal{I}}\left\{S^{im}_{l0} + \sum_{1\le t'\le t}\left(B^{im}_{lt'} + s^{im}_{lt'}\right)\right\}\right\} -$$
$$\min\left\{R^0_{l0} + \sum_{1\le t'\le t-1}\left(A^0_{lt'} + r^0_{lt'}\right), \min_{i\in\mathcal{I}}\left\{S^{im}_{l0} + \sum_{1\le t'\le t-1}\left(B^{im}_{lt'} + s^{im}_{lt'}\right)\right\}\right\}. \quad (5.15)$$

Alternatively, b_{lt} can be written as

$$b_{lt} = \min\left\{R^0_{l0} + \sum_{1\le t'\le t}\left(A^0_{lt'} + r^0_{lt'}\right), \min_{i\in\mathcal{I}}\left\{S^{im}_{l0} + \sum_{1\le t'\le t}\left(B^{im}_{lt'} + s^{im}_{lt'}\right)\right\}\right\} - \sum_{t'=1}^{t-1} b_{lt'}.$$

With the node set in place, we define arcs between the nodes. The first set of arcs connect the initial serviceable system inventory nodes to the corresponding final serviceable system inventory nodes. For a serviceable system starting in state n and a planning horizon T, the arc will terminate at the final serviceable system inventory n' for which $\psi^{n-n'}_{l0} \ge T$ and $\psi^{n-n''}_{l0} < T\ \forall\ n'' > n'$. If a system will be fully used prior to the end of the planning horizon, the arc will terminate at the 0 system state node.

The flow of unused assembly inventory is denoted by arcs between adjacent assembly inventory nodes. Flow over an arc between an assembly inventory node and an assembly node in the same time period occurs if systems are assembled in that time period. These assembly arcs will have upper and lower bounds equal to U_{lt} and Q_{lt}. The network will also contain an arc, with no upper or lower bounds, from the final assembly inventory node to the sink. From the assembly nodes, there are arcs to the state specific assembly nodes in the same time period. For a state specific assembly node n in period t, this arc will have a lower bound equal to Q^n_{lt}.

From each state specific assembly node for location l and state n in period t, there will be arcs to the final serviceable system inventories at location l and to other locations l'. The termination point for these arcs depends upon the assembly time at location l, γ_{lt}, the transit time, $\nu_{ll'}$, and the usage rate at the final destination,$\psi^n_{l't}$. Arcs from the state specific assembly nodes at location l to final serviceable system inventories also at location l will terminate at the highest state n' for which $\psi^{n-n'}_{l(t+\gamma_{lt})} \ge$

$T-(t+\gamma_{lt})$. For the arcs going to another location l', the arcs will terminate at the highest final serviceable system inventory for which $\psi^{n-n'}_{l'(t+\gamma_{lt}+\nu_{ll'})} \geq T-(t+\gamma_{lt}+\nu_{ll'})$.

From each final serviceable system inventory node, there will be exactly one arc connecting it to the corresponding target serviceable system inventory node. These arcs do not have capacities but rather are the only arcs in the network with costs. Specifically, they will have a convex cost function $\mathfrak{G}^{0n}_{lT}(\overline{S^{0n}_{lT}}-\mathbf{S^{0n}_{lT}})$ where $\mathbf{S^{0n}_{lT}}$ is the flow across the arc. While we allow $\mathfrak{G}^{0n}_{lT}(\cdot)$ to be any convex function, we can transform the problem into a network flow problem with a linear cost funciton by adding parallel arcs between the final and target inventories, using $\mathfrak{G}^{0n}_{lT}(\cdot)$ to set the arc flow costs and the capacities on each of the parallel arcs. The full explanation behind the method for transforming convex cost flows can be found in Ahuja et al. [6]. Target serviceable system inventory nodes will have a single outgoing arc to the sink node with no costs or capacities.

A final set of arcs represents the flow of reparable inventory from one location to another. These arcs will begin at an assembly inventory node at some location l in period t, will terminate at another location l' in period $t+\nu_{ll'}$ and will have an upper bound of $\mu^{0r}_{ll't}$.

With this network structure, we can relate arc flows to the original decision variables and constraints of the system assembly submodel. Flow between assembly inventory nodes at the same location equates to the conservation of inventory variables $\mathbf{R^0_{l(t-1)}}$ and $\mathbf{S^{im}_{l(t-1)}}$. Flow between assembly inventory nodes at different locations represent the $\mathbf{r^0_{ll't}}$ and $\mathbf{r^0_{ll't'}}$ variables in the reparable systems inventory constraints (5.2), and the capacity for reparable systems flow between locations, constraints (5.11). Arc flows from the assembly inventory to the system assembly nodes equate to the total number of systems entering assembly at a given time period, $\sum_{n\in\mathcal{N}}\mathbf{y^{0n}_{lt}}$, which equals $\mathbf{y^{im}_{lt}}$ for each i due to the specially constructed source value described above. Individual $\mathbf{y^{0n}_{lt}}$ variables are represented by the flow between system assembly nodes and the state specific assembly nodes. Movement of serviceable systems between locations, $\mathbf{s^{0n'}_{ll't(t+\gamma_{lt})}}$ and $\mathbf{s^{0n'}_{l'lt}}$, is captured by the flow on the arcs between state specific assembly nodes and the final serviceable systems inventory nodes. The artificial build and bal-

ance variables $\mathbf{w_{lt}^{imn}}$ are not needed because of the single state simplifying assumption and the corresponding source values. Restriction of serviceable systems shipment only upon completion of assembly and test is captured by the shipment arcs out of the state specific assembly nodes.

Note that this network flow formulation does not consider the capacity constraint for serviceable systems flow, (5.12). If there are no state specific assembly requirements (i.e. $Q_{lt}^n = 0$), then the capacity constraint for serviceable systems flow can be included by replacing the state specific system assembly nodes with location specific system assembly nodes and adding an upper bound on the incoming arc from the system assembly node. Numerous arcs will flow out of a location specific system assembly node denoting the different system states that are assembled and end up in the appropriate final serviceable systems inventory. Recall that the planning horizon is very short and that we consider only the serviceable system inventory at the end of the planning horizon. Thus systems assembled during the planning horizon will flow directly to the final serviceable inventory nodes and not intermediary serviceable inventory nodes. An example network is illustrated in Figure 5-5. These relationships yield the following result.

Lemma 5.2.1 *If all of the modules are in the same state and there are either state specific proficiency requirements or serviceable system transportation capacities, there exists a one-to-one relationship between a solution to the system assembly submodel and the network flow problem.*

The assumption that all modules are in the same state can be relaxed and still retain the efficient structure of the problem. In particular, if all modules are at or above a given state m, and all systems are assembled in a state $n \leq m$, then the same network flow representation can be used. This observations leads to an iterative heuristic algorithm, an example is described below, that solves multiple network flow problems. We first solve the problem as a network flow with one module state. Then, holding the module use, system assembly, and transportation decisions as input, we solve a second network flow problem with a second module state. We continue in this

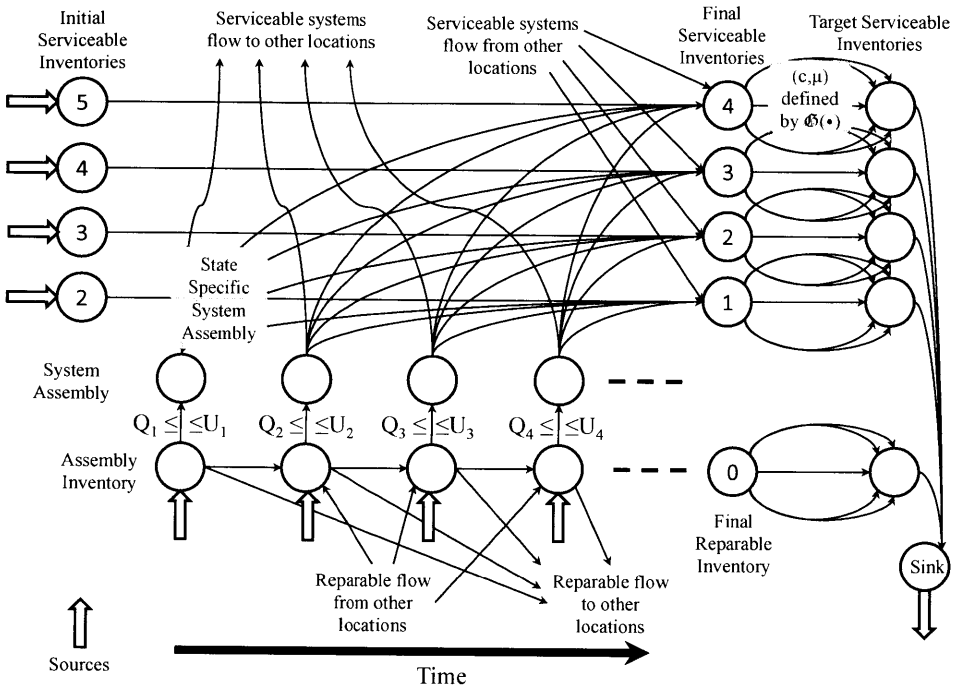

Figure 5-5: Generic System Submodel Network Flow Representation

manner until we have exhausted all possible system states. By solving a sequence of network flow problems, the solution quality can be improved relative to solving a single network flow formulation. We implemented this approach choosing the system states in order from largest to smallest.

As an example, consider an algorithm that begins by solving the network flow problem for the maximum possible system state, $\widehat{n}$. In this iteration, the assembly sources described in equations (5.14) and (5.15) will be based on the corresponding serviceable module inventories that have a high enough state to be used for system assembly (i.e., $m \geq \widehat{n}$). In addition, the target serviceable inventories considered will be for only state $\widehat{n}$. The resulting solution will include assembly and transshipment decisions which are then used to update the test stand capacity constraints, the shipment capacity constraints, the reparable system inventories, and the serviceable module inventories. The next iteration will proceed in the same manner but will consider system assemblies in state $\widehat{n}-1$ with updates to the inventories and constraints.

Iterations continue until all possible system assembly states have been considered. Combining the solutions from each iteration provides a comprehensive assembly and transhipment plan that can be implemented or used as part of policy tradeoffs.

While such an algorithm is not guaranteed to solve the system assembly subproblem optimally, computational tests indicate that this algorithm finds good solutions in a fraction of the time required to solve the integer programming formulation. For a number of small and mid-sized instances (5 locations, 10 system states, and 50 modules) for which the integer formulation could be solved to optimality, the iterative algorithm produced solutions within 25% of the optimal solution with many instances within 10% and an average of 15%. For these instances, between 20 and 150 seconds were needed to find the optimal integer solution while the iterative algorithm provided a solution in less than a quarter of a second. For some larger instances (10 locations, 10 system states, 50 modules, and time varying assembly capacity), it was possible to fully meet the serviceable inventory and in every case the iterative network flow algorithm found an optimal solution in less than a third of a second. In all of these cases, the linear programs took over a minute to solve and the integer program did not solve optimally in 10 minutes for some instances. In one case, the best integer programming solution found after 10 minutes had a cost of 19 even though the LP had a tight lower bound of 0. The iterative network flow algorithm as well as the integer programming formulation were implemented in AMPL with CPLEX 11.21.

Algorithms that require more iterations could be used that select the assembly state and location based on the structure of $\mathfrak{G}_{lT}^{0n}(\cdot)$ (i.e., pick the sequence of assembly states, locations, and target inventories based on the changes in slope of $\mathfrak{G}_{lT}^{0n}(\cdot)$). For instance, the location and system state with the largest decrease in cost could be iteratively selected and network flow instances constructed to meet those demand. In Section 5.4 we discuss another efficient planning decomposition of the problem that exploits this network flow formulation of the problem.

5.2.2 Solving the System Submodel: A Greedy Algorithm for a Single Uncapacitated Location

Recall that the system assembly submodel seeks to assemble systems to match the serviceable targets provided by the planning model subject to reparable system and serviceable module inventories, build and balance constraints, and a convex objective function. Module repair decisions are treated as input as they are made in the planning model and module repair submodel. In the previous section we addressed the difficulties associated with the build and balance constraints by assuming all serviceable modules are in the same state. In this section we will examine the single location problem when there are no upper or lower bounds on system assembly. For ease of explanation, assume the assembly time is 0, that is γ_{lt}=0. We will relax this assumption subsequently. As the problem is uncapacitated, we can assemble all systems in the last time period. As mentioned previously, the objective is to minimize:

$$\sum_{n \in \mathcal{N}} \mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - \mathbf{S_{lT}^{0n}}),$$

for the specified $\mathfrak{G}_{lT}^{0n}(\cdot)$ convex cost functions and the end of horizon target serviceable system inventories obtained from the planning model, $\overline{S_{lT}^{0n}}$. The ending inventory of systems in each state n depend on the reparable system and serviceable module inventories. As a reminder, exactly one reparable system is needed to assemble a serviceable system and the reparable systems have no impact on the ending state of a system. In addition, for each serviceable system that is assembled, we need exactly one serviceable module of each type, with the system state upper bounded by the state of the module with the lowest state. We define the set $Active \subseteq \mathcal{N}$ to be the set of system states that can feasibly be constructed given the current serviceable module inventory. Initially, *Active* is the set of all states between the minimum state and the highest state n for which all modules types have at least one module in state n or higher. Using the prior notation, $Active = \{\text{argmin}\, n \in \mathcal{N}, \ldots, \text{argmax}\, n \in \mathcal{N} : \sum_{m \geq n} \mathbf{S_{IT}^{im}} \geq 1 \; \forall \; i \in \mathcal{I}\}$. If there are

no serviceable modules of a given type, *Active* is defined to be the null set.

The greedy algorithm builds systems by picking the system state from *Active* that reduces the cost of the current solution by the most and assembling one system in that state with modules in the minimum possible state. This continues until all feasible target system inventories have been matched, the reparable system inventory has been depleted, or there are no longer any feasible system assembly decisions. The set *Active* will be updated throughout the algorithm each time the highest state serviceable inventory of a module is depleted. We initially set the number of serviceable systems in state n, $\mathbf{S_{lT}^{0n}}$, equal to 0.

Greedy Algorithm for System Assembly Submodel w/o Capacity Constraints

While $|Active| > 0$, $\mathbf{R_{lT}^{0}} > 0$, and $\max_{n \in Active} \mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - \mathbf{S_{lT}^{0n}}) > 0$:

1. Let $\widetilde{n} = \text{argmax}_{n \in Active}\{\mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - \mathbf{S_{lT}^{0n}}) - \mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - (\mathbf{S_{lT}^{0n}} + 1))\}$
2. Assemble one system in state $\widetilde{n}$.
   (a) Reduce $\mathbf{R_{lT}^{0}}$ by one
   (b) For all $i \in \mathcal{I}$, reduce $\mathbf{S_{lT}^{im}}$ by one for the minimum $m \geq \widetilde{n}$ for which $\mathbf{S_{lT}^{im}} > 0$.
   (c) Increment $\mathbf{S_{lT}^{0\widetilde{n}}}$ by one.
3. Update *Active* if $\mathbf{S_{lT}^{im}}$ is depleted for the maximum state serviceable inventory for some module i.

We begin by showing that this algorithm optimally solves the single location, uncapacitated system assembly problem with zero assembly times. To do so, we first introduce an inventory swap procedure we will use as part of the proof. This inventory swap procedure ensures that, whenever the system assembly solution is updated, the remaining systems are assembled with the modules in the minimum possible state. The procedure begins at a given state and works upward, iteratively releasing serviceable systems from the inventory and rebuilding them with modules in the minimum possible state.

Inventory swap procedure: Begin with a lower bound ($\underline{n}$) system state and let $n = \underline{n}$.

1. If $n \in$ *Active*:
   (a) Release the system from the n state serviceable systems inventory that has the highest minimum state module.
   (b) Return the reparable system and the serviceable modules from the released system to their respective inventories.
   (c) Assemble a system in state n with the minimum possible state components from the serviceable module inventory (i.e. those components in the serviceable inventory with a state greater than or equal to n for which there are no lower state serviceable modules that can be used to build a system in state n).
   (d) Place the new serviceable system in the state n serviceable systems inventory.
   (e) Update *Active* and update n equal to $n' > n : \mathbf{S_{lT}^{0n'}} > 0$, and repeat.
2. If not, terminate.

With this procedure in place, we can now prove the optimality of the greedy algorithm.

Theorem 5.2.2 *The greedy algorithm optimally solves the single location, uncapacitated system submodel with zero assembly times.*

Proof : Consider the contribution of each system state n to the overall objective function value. If the greedy algorithm does not optimally solve the problem then there exists at least one serviceable systems inventory in the optimal solution, S^* that differs from the greedy solution $\mathbf{S}$. Beginning with the lowest state $n \in \mathcal{N}$ at which they differ, there are three cases for the serviceable systems inventory:

1. $S_{lT}^{*0n} < \mathbf{S_{lT}^{0n}}$: Reduce $\mathbf{S_{lT}^{0n}}$ by one and return the reparable system and serviceable modules from the system with the highest minimum component to their

respective inventories. The original greedy solution is obviously feasible as is the updated solution. By the convexity and unique minimum properties of $\mathfrak{G}_{lT}^{0n}(\cdot)$, $\mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - S_{lT}^{*0n}) \geq \mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - (\mathbf{S_{lT}^{0n}} - 1)) > \mathfrak{G}_{lT}^{0n}(\overline{S_{lT}^{0n}} - \mathbf{S_{lT}^{0n}})$. This implies that the cost of the updated greedy solution has increased. As S^* is the optimal solution, however, there must exist a higher state n' for which $S_{lT}^{*0n'} > \mathbf{S_{lT}^{0n'}}$ and $\mathfrak{G}_{lT}^{0n'}(\overline{S_{lT}^{0n'}} - S_{lT}^{*0n'}) < \mathfrak{G}_{lT}^{0n'}(\overline{S_{lT}^{0n'}} - \mathbf{S_{lT}^{0n'}})$. This higher state will be addressed in one of the next two cases. Finally, perform an inventory swap procedure beginning at the minimum state n for which $\mathbf{S_{lT}^{0n}} > 0$.

2. $S_{lT}^{*0n} > \mathbf{S_{lT}^{0n}}$ and $\mathbf{S_{lT}^{0n'}} > 0$ for some $n' > n$: In this case we need to increase the number of serviceable systems in state n to match the optimal solution. This can be performed by pulling a reparable system and serviceable modules from inventory and assembling the system in state n or, if that is not possible, by downgrading a higher state system. In the later case, at some point during the greedy algorithm, when the last system in state n' was assembled, the marginal benefit of building a system in n' and n were compared. As n' was chosen, we know that the marginal benefit was higher for one system in state n' versus one system in state n. By the convexity of $\mathfrak{G}_{lT}^{0n}(\cdot)$, specifically the nondecreasing slope property, this implies that the marginal benefit of an additional system in n is less than the marginal cost of a downgrading a unit in n'. Thus, the updated solution obtained by downgrading the lowest minimum module state system currently in state n' to state n' will have a higher cost.

   In the case where we pull modules from inventory to assemble the new system in state n, we know that the modules must not have been in the inventory at the end of the greedy algorithm else the greedy algorithm would have assembled the system in state n. Accordingly, the modules in the inventory must have been freed up by an inventory swap procedure when the number of systems in some lower state, $\widetilde{n}$, was reduced. When the system that freed up modules now used in the last state n system was assembled, the greedy algorithm compared the marginal benefit of all possible states that could be built

with those modules. Since the algorithm chose the lower state, n', we know that the marginal cost of reducing the serviceable inventory for n' by one is higher than the marginal benefit of increasing the serviceable inventory for state n by one. As the modules are now available for use in state n, the minimum state module was released by some lower state during the inventory swap procedure. By repeating the above argument with n' and the next lower state updated in the inventory swap procedure, we will continue to move further down the serviceable systems inventories until we arrive at $\widetilde{n}$, the serviceable inventory that was reduced in step 1 above. By combining all of the resulting inequalities, we see that the marginal cost decrease of increasing the serviceable system inventory for state n by one is less than the marginal cost increase incurred when we decreased the serviceable system inventory for state $\widetilde{n}$. Both solutions are feasible but the updated solution has a higher cost.

3. $S_{lT}^{*0n} > \mathbf{S_{lT}^{0n}}$ and $\mathbf{S_{lT}^{0n'}} = 0$ for all $n' > n$: As with the previous case, we assemble a system in state n with modules from the inventory. By the same reasoning as above, the cost decrease from this additional system is less than the cost increase from the reduction of a serviceable system inventory with a lower state that made the modules available. The remaining question is if such an assembly is possible. As the steps above have been completed for all lower level inventories, we know that the number of serviceable systems in the updated solution equals the number in the optimal solution for each of those states. In addition, the greedy algorithm combined with the inventory swap procedure ensures that these inventories are comprised of the minimum possible module. Since the optimal solution contains an additional system in this state, it must also be feasible under the updated solution.

By repeating the steps above and updating the greedy solution system inventories to match the optimal solution system inventories in increasing state order, we see that each update increases the cost of the solution. This implies that the optimal solution has a cost higher than the greedy solution but as both are feasible, this is

a contradiction of optimality. Accordingly, the greedy solution must build the same number of systems in the same states as the optimal solution.

■

Based on the optimality of the greedy algorithm for zero assembly times, we can now extend the result to nonzero assembly times.

Corollary 5.2.3 *The modified greedy algorithm is optimal for nonzero assembly times.*

Proof : Since there are no capacity constraints, we only need to consider the assembly of systems in a single time period and Lemma 5.2.2 shows that the greedy algorithm is optimal for choosing which systems to assemble in a single time period. Consider the latest period $\widehat{t}$ for which $\widehat{t} + \gamma_{l\widehat{t}} \leq T$. Any time periods t' for which $t' + \gamma_{lt'} > T$ will not contribute to the objective function and thus do not need to be considered. For any t' for which $t' + \gamma_{lt'} \leq T$ and $t' < \widehat{t}$, $\mathbf{S}^{\mathbf{im}}_{\mathbf{lt'}} \leq \mathbf{S}^{\mathbf{im}}_{\mathbf{l\widehat{t}}}$ $\forall\ i \in \mathcal{I}$ and $m \in \mathcal{M}$ as additional serviceable modules may become available later in the time horizon. The same is true for the reparable systems inventory. Accordingly, any solution that is attainable at period t' is also attainable at period $\widehat{t}$. Thus, the greedy algorithm applied in period $\widehat{t}$ is optimal. ■

As a final note, it is highly likely that there will be multiple optimal solutions in which modules are used differently to assemble systems. While the greedy solution builds the optimal number of systems in each state, it may not match the optimal solution from an MIP solver on a module by module basis. However, it is possible to use the greedy algorithm to update a solution to the system assembly problem if there are module concerns that are not addressed in the formulation. For instance, it might be desirable to build systems in which the minimum module is as far from the system state as possible due to failure concerns. Taking the final system inventories as an input, such a problem can again be solved by a greedy algorithm by starting with the highest states and progressing to the lower states using the highest possible modules to build each system. Other module concerns can be addressed similarly using the desired serviceable systems inventories as an input. In general, there will be

multiple optimal solutions that assemble the correct number of serviceable systems but with different modules. Side constraints not considered as part of the system assembly model can be used to update an optimal system assembly solution to match operational requirements.

5.2.3 Solving the System Submodel: Linear Underage and Overage Cost Functions for a Single Location

Recall that the system assembly submodel seeks to assemble systems to match the serviceable targets provided by the planning model subject to reparable system and serviceable module inventories, build and balance constraints, and a convex objective function. Module repair decisions are treated as input as they are made in the planning model and module repair submodel. While the previous two sections have made assumptions about the constraints of the problem, this section will focus on the cost functions. By assuming a specific form for the cost function, we can optimally solve the system assembly subproblem with capacity constraints. Consider a special case of the single location system assembly subproblem in which the convex cost functions $\mathfrak{G}_{lT}^{0n}(\cdot)$ are linear below and above the target (say a V centered at the target) and that the slope is non-increasing in the system state. While this form of the cost function does not capture the reduction in marginal value of additional systems, it is an intuitive form for the cost function as having too few high state systems is worse than having too few low state systems. In addition, having too many high state systems gives us more flexibility than having too many low state systems. We will also assume that there are no state specific maintenance proficiency constraints (i.e., $Q_{lt}^{n} = 0$ for all systems states n). The greedy algorithm can be updated when $Q_{lt}^{n} > 0$, although the analysis is more complicated. Of note is the fact that this algorithm explicitly exploits the fact that the system assembly planning horizon is very short and assumes that a system assembled in a given state will remain in that same state over the entire planning horizon.

Greedy Algorithm for System Assembly Submodel w/ Linear Underage

and Overage Costs

1. Beginning with the highest state and first time period, assemble as many systems as possible respecting the modular and system inventories, the capacity constraints, and the system state target (dont build more systems than the target). Use modules with the minimum possible state when constructing a system

2. Continue assembling systems moving later in time as needed due to reparable system and serviceable module inventories and test stand capacity constraints.

3. Once a particular system state is evaluated in every time period (due either to capacities, inventory, or reaching the system target), move to the next lower state and repeat.

4. Once an initial build is complete for all states, beginning in the first time period and moving forward in time, delay system assembly, for the lowest possible system state, to later periods as necessary to meet the overall maintenance proficiency constraints. If proficiency constraints cannot be met by shifting system assembly to a given time period, assemble systems in that time period to the highest state possible to meet the proficiency constraints.

To show that this greedy algorithm optimally solves the single location system assembly subproblem with the special cost function described above, Lemma 5.2.4 will first show that it finds a feasible solution if such a solution exists. Lemma 5.2.5 then shows that the solution produced by the greedy algorithm has the minimum possible cost at the end of step 3. Finally Lemma 5.2.6 shows that any additional increase in cost incurred in step 4 is the minimum possible increase. Consequently, the greedy solution has the same cost as the optimal solution.

Lemma 5.2.4 *The greedy algorithm for the system assembly submodel with linear costs finds a feasible solution if one exists.*

Proof : By construction, the algorithm satisfies the reparable systems (5.2) and serviceable module (5.7) inventory constraints, the test stand constraints (5.4), the build constraints (5.8), and the balance constraints (5.9). As we only consider a single location, the transportation constraints are not considered. Accordingly, the only constraints we need to consider are the overall (5.5) maintenance qualification constraints. Let t be the earliest period in which the overall maintenance qualification constraint is not met in the final schedule. The last step in the algorithm attempts to meet the proficiency constraint by either shifting or assembling additional systems. As an assembly decision can always be shifted later, if we are unable to shift any assembly decision this implies all periods prior to t satisfy their proficiency constraints with equality. In such a case, the algorithm will attempt to assemble systems at period t to meet the proficiency constraint at period t. If, however, it can not assemble enough systems in that period to meet the proficiency constraints, this implies there are not enough modules available of at least one type by period t to meet the cumulative proficiency constraints for periods $\{1,2,\ldots,t\}$. As each system requires only one of each module this implies no solution exists that meets the proficiency constraints due to the lack of modules. ■

Having shown that the greedy algorithm finds a feasible solution, if one exists, we next consider the quality of the solution. We begin by showing the the solution at the end of step 3 of the algorithm provides the minimum value that can be attained based on the inventory and test stand capacity constraints.

Lemma 5.2.5 *At the end of step 3, the greedy solution will generate the same or lower cost as any other solution that ignores the maintenance proficiency constraints.*

Proof : As we are only considering a single location, the objective function can be broken down by the contribution from each system state, which is a function of the number of systems that are in that state at the end of the planning horizon. Consider the number of systems in the highest system state at the end of step 3. If the greedy algorithm builds exactly the target number, this is the minimum contribution for the highest system state. Thus any other solution's contribution can only be the same

or higher. The algorithm will build as many of the highest state systems as possible, without going over the target. If the greedy algorithm does not build up to the target level, we know that the any other solution that ignores the proficiency constraints must build the same or or a smaller number of systems ending in this state. In this case, as the cost function is linear and decreasing in the number of systems up to the target, any other solution must have the same or higher cost contribution as generated by the greedy algorithm. This shows that the cost contribution at the end of step 3 for the highest system state is the same or lower for the greedy solution as it is for any other solution that ignores the proficiency constraints.

Consider the number of systems in any lower system state found in the greedy solution and found in any other solution that ignores the proficiency constraints. If, for some state, the other solution builds fewer systems than found in the greedy solution, then the cost contribution will be higher since the greedy solution will only build up to the target. Since the greedy solution attempts to build as many systems as possible, up to the target, another solution can only build more systems than the greedy solution if it builds fewer systems in a higher state. It then uses those modules to build a lower state system. Or, the greedy solution is at the target, and the other solution builds more systems than the target. In the second case, the other solution will have a higher cost since the cost function has a unique minimum at the target. Recall that the cost functions are non-increasing for a given argument as a function of the system state. Due to this property, in the first case we know that the decrease in cost for the additional lower level state systems is less than the corresponding decrease in cost if higher state systems were built. Accordingly, at the end of step 3, the greedy solution will have the same or lower cost as any other solution that ignores the proficiency constraints. ■

To meet the proficiency requirements we can either shift assembly decisions to a later period or assemble systems in addition to those assembled as a consequence of executing the first three steps of the algorithm. As shifting assembly to later periods does not affect the objective value, we will then show that any additional assemblies that are accomplished to meet the proficiency constraints are the minimum amount

possible.

Lemma 5.2.6 *Any cost incurred as a consequence of additional assembly decisions made when executing step 3 is the minimum possible.*

Proof : Consider period t, the first period that the proficiency constraint is not met after the completion of step 3. If the proficiency constraint can be met by shifting an assembly event to a later period, then the greedy algorithm will shift assembly actions to period t, which does not affect the objective function. However, if all periods prior to t exactly meet their proficiency requirements, the only way meet the proficiency requirement in period t is to assemble enough systems in that period to meet the proficiency requirement. We can assume that all system states that can be feasibly assembled in period t are at their target level. If they were not, then by the argument above, the algorithm would have assembled more of the systems in order to minimize the objective by meeting the target level. In step 4 of the algorithm, the highest possible state systems are assembled to augment the solution in order to meet the proficiency requirements. As the value of $\mathfrak{G}_T^{0n}(\cdot)$ is defined to be 0 at the targets and increase for each additional system assembled above the target levels, the choice of system states to assemble must be chosen to incur the minimum increase in cost. Since the slope of the state dependent objective functions are non-increasing in the system state, $\mathfrak{G}_T^{0n}(S+1) - \mathfrak{G}_{lT}^{0n}(S) \leq \mathfrak{G}_T^{0n'}(S'+1) - \mathfrak{G}_T^{0n'}(S')$ for all $n' < n, S \geq \overline{S_T^{0n}}$, and $S' \geq \overline{S_{lT}^{0n'}}$. Accordingly, by assembling a system in the highest state possible (i.e., largest n such that there exists $m \geq n$ for which $S_t^{im} > 0$) at period t, the objective function value will increase by $\mathfrak{G}_{lT}^{0n}(S+1) - \mathfrak{G}_{lT}^{0n}(S)$, the minimum amount possible. ■

Combining these three lemmas yields the following result.

Theorem 5.2.7 *The assembly schedule provided by the greedy algorithm has the same cost as the corresponding optimal solution.*

5.3 Module Repair Submodel

Having addressed the system assembly submodel we turn our attention to module repair. Unlike the system assembly problem in which the assemble state of a system did not impact the assembly time, the repair decisions in the module submodel have a direct impact on how long a module spends in maintenance. This drives two specific changes to the module repair submodel as compared to the system assembly submodel. The first of these is the introduction of a decision horizon that is smaller than the planning horizon. We let Γ denote the end of the maintenance decision horizon while T is the end of the module repair planning horizon. Due to the time spent in maintenance, the planning horizon for the module submodel is longer than that of the system assembly submodel. However, the repair time for a module can vary significantly depending on the level of repair. Thus, in the module repair submodel we will make maintenance decisions over the horizon $\{1, \ldots, \Gamma\}$ while we evaluate the impact of those decisions over $\{\Gamma, \ldots, T\}$.

A second change pertains to shipping decisions. Serviceable modules can only be shipped coming out of repair during periods $\{\Gamma + 1, \ldots, T\}$. This limits the optimization procedure from moving modules between locations solely to gain objective value. One item of note in this formulation is the transition of the variables $\mathbf{y}_{\mathbf{lt}}^{\mathbf{im}}$ into data elements $\underline{y_{lt}^{im}}$. This is a direct result of the fact that, in any period, we will first solve the system assembly problem. This solution will result in assembly decisions which in turn draw inventory from the serviceable module inventories. These serviceable module demands are then treated as data in the module submodel. Accordingly, for each module type $i \in \mathcal{I}$ we have the following integer program.

The remainder of the variables are defined the same as in the planning model formulation:

- $\mathbf{R}_{\mathbf{lt}}^{\mathbf{im}}$ the reparable module inventory,
- $\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm'}}$ the number of reparable modules of type i currently in state m to repair to state m' at location l beginning in period t

- $\mathbf{r^{im}_{ll't}}$ the number of reparable systems to ship from location l to location l' in period t, and
- $\mathbf{s^{im}_{ll't}}$ the number of serviceable systems in state n to ship from location l to location l' in period t.

Minimize weighted difference from planning model targets

$$\text{minimize} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} \sum_{\Gamma < t \leq T} \mathfrak{G}^{im}_{lt} (\overline{S^{im}_{lt}} - \mathbf{S^{im}_{lt}}) \tag{MSmP}$$

subject to

Reparable module inventory: inventory from previous period, plus modules coming out of service and inflow from other locations, minus modules placed in repair, plus net inflow of reparable modules

$$\begin{aligned} \mathbf{R^{im}_{lt}} = \mathbf{R^{im}_{l(t-1)}} + A^{im}_{lt} + r^{im}_{lt} - \sum_{\substack{m' \in \mathcal{M}: \\ m' \geq m}} \mathbf{x^{imm'}_{lt}} + \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \nu_{l'l} = t}} \mathbf{r^{im}_{l'lt'}} - \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{r^{im}_{ll't}}, \\ \forall\, m \in \mathcal{M}, l \in \mathcal{L}, \quad 1 \leq t \leq \Gamma, \end{aligned} \tag{5.16}$$

Serviceable module inventory: inventory from previous period, plus modules coming out of repair and inflow from other locations, minus modules used in system assembly, plus modules coming out of repair, plus net inflow of reparable modules

$$\begin{aligned} \mathbf{S^{im}_{lt}} = \mathbf{S^{im}_{l(t-1)}} + B^{im}_{lt} + s^{im}_{lt} - \underline{y^{im}_{lt}} + \sum_{\substack{m' \in \mathcal{M}: \\ m' \leq m}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \tau^{im'm}_{lt'} = t}} \mathbf{x^{im'm}_{lt'}} + \\ \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \nu_{l'l} = t}} \mathbf{s^{im}_{l'lt'}} - \sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{s^{im}_{ll't}}, \\ \forall\, m \in \mathcal{M},\ l \in \mathcal{L}, \quad 1 \leq t \leq T, \end{aligned} \tag{5.17}$$

Module repair capacity

$$\sum_{m \in \mathcal{M}} \sum_{\substack{m' \in \mathcal{M}: \\ m' \geq m}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \tau^{imm'}_{lt'} > t}} \mathbf{x^{imm'}_{lt'}} \leq M^{i}_{lt} - \sum_{m \in \mathcal{M}} \sum_{t < t' \leq T} B^{im}_{lt'}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.18}$$

Reparable module flow between locations

$$\sum_{m \in \mathcal{M}} \mathbf{r}^{\mathbf{im}}_{\mathbf{ll't}} \leq \mu^{ir}_{ll't}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l', \quad 1 \leq t \leq T, \tag{5.19}$$

Serviceable module flow between locations

$$\sum_{m \in \mathcal{M}} \mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}} \leq \mu^{is}_{ll't}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l', \quad 1 \leq t \leq T, \tag{5.20}$$

Aggregate module flow between locations

$$\sum_{m \in \mathcal{M}} \left(\mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}} + \mathbf{r}^{\mathbf{im}}_{\mathbf{ll't}}\right) \leq \mu^{i}_{ll't}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l', \quad 1 \leq t \leq T, \tag{5.21}$$

Shipment of serviceable modules after decision horizon only allowed when coming out of repair

$$\sum_{\substack{l' \in \mathcal{L}: \\ l' \neq l}} \mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}} \leq \sum_{\substack{m' \in \mathcal{M}: \\ m' \leq m}} \sum_{\substack{1 \leq t' \leq t: \\ t' + \tau^{im'm}_{lt'} = t}} \mathbf{x}^{\mathbf{im'm}}_{\mathbf{lt'}}, \qquad \forall\, m \in \mathcal{M},\ l \in \mathcal{L}, \quad \Gamma < t \leq T, \tag{5.22}$$

Maintenance decisions allowed only during decision horizon

$$\mathbf{x}^{\mathbf{imm'}}_{\mathbf{lt}} = 0, \qquad \forall\, m, m' \in \mathcal{M},\ l \in \mathcal{L}, \quad \Gamma < t \leq T \tag{5.23}$$

Nonnegativity and integrality of the decision variables

$$\mathbf{R}, \mathbf{S}, \mathbf{x}, \mathbf{r}, \mathbf{s} \in \mathbb{Z}^{+}. \tag{5.24}$$

While the formulation remains an integer program, the number of maintenance events for each module type in the planning horizon is sufficiently small so that the problem can be solved using a commercial integer programming package. This is a direct result of the fact that modules depend upon independent resource capacity. In computational tests, a number of instances with 10 module states, 10 operating

locations, plus a depot, and a planning horizon of 90 periods solved to optimality in less than a minute.

5.3.1 Stochastic Module Degradation

In contrast to the system submodel, random failures directly impact the module submodels through the module inventories and the repair constraints. As with the system submodel, the module submodels will take into account the past random occurrences as input for the inventories and modular capacity constraints through initial condition data. In addition, as the module submodels will be solved after the system submodel, the assembly decisions from the system submodel will also be used in the module submodels as initial conditions. Suppose we consider stochastic module degradation, that is, suppose that modules may not be in the state corresponding to the reparable inventory they are assigned to. In this scenario, the actual state of the module is discovered when it is placed in repair and the technician performs an initial inspection.

Since we have modeled maintenance as an activity that requires one unit of resource over a given horizon, random module state changes do not consume more resources in a single time period but rather the time horizon over which a module stays in maintenance. From a modeling and implementation standpoint, the question then becomes the method by which stochastic module states are treated in the optimization problem. One potential option is to ensure that the maintenance capacity is not exceeded in expectation. Another is to ensure that the maintenance capacity is not exceeded with high probability with a framework such as robust optimization. In this section we will take a combined approach in which the optimization problem ensures that, from a planning standpoint, the maintenance capacity constraints are not violated in expectation. During schedule execution, we will use information about modules currently in maintenance, along with characteristics of the degradation process, to augment the maintenance capacity if needed. This augmentation could come in the form of manpower, longer shifts, or additional serviceable inventory.

Planning To generalize the formulation above, we introduce coefficients $a_{ltt'}^{imm'}$ in the

maintenance capacity constraints. These coefficients denote the percent of modules that were in reparable module inventory in state m that we chose to repair to state m' at location l beginning in period t' and are still in repair at period t. For the deterministic case these coefficients are defined as:

$$a_{lt't}^{imm'} = \begin{cases} 1 & \text{if } t > t' \text{ and } t' + \tau_{lt'}^{imm'} > t, \\ 0 & \text{otherwise.} \end{cases}$$

The functional form of the deterministic a coefficients is represented by the solid line in Figure 5-6. The maintenance capacity constraints can thus be rewritten as:

$$\sum_{m \in \mathcal{M}} \sum_{\substack{m' \in \mathcal{M}: \\ m' \geq m}} \sum_{1 \leq t' \leq t} a_{lt't}^{imm'} \cdot \mathbf{x_{lt'}^{imm'}} \leq M_{lt}^{i} - \sum_{m \in \mathcal{M}} \sum_{t < t' \leq T} B_{lt'}^{im}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T. \tag{5.25}$$

When we consider stochastic module degradation, the a coefficients will change based on the distribution of actual module states in each reparable module inventory. If the technician finds a module that has less degradation than expected, the a coefficients may be lower than the corresponding deterministic coefficients. This implies the modules will leave the maintenance repair facility earlier than expected. In other cases, the a coefficients may be higher than the deterministic coefficients due to modules being in a lower state than expected. In either case, however, it is unlikely that every module that comes in for maintenance will differ from the expected state. In fact, most modules should be in the state we expect unless there is a significant bias in our degradation estimates for usage based modules.

Another reasonable assumption is that the actual state of a module beginning repair does not imply that other modules of the same type in the reparable inventories are in states different from their expected state. This independence assumption between modules, of both the same and different states, is again based on an unbi-

ased estimate of usage patterns. Based on these insights, the a coefficients will likely represent an S curve shown by the dashed line in Figure 5-6.

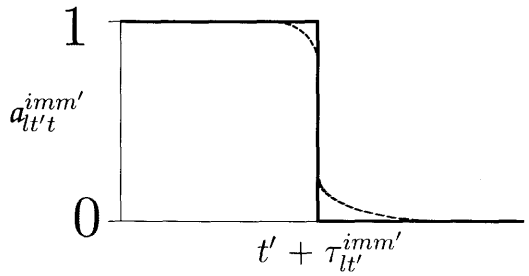

Figure 5-6: Sample $a_{lt't}^{imm'}$ values for deterministic and stochastic degradation.

By characterizing the stochastic nature of the a coefficients, constraints (5.25) can be used in the planning model to ensure that the maintenance capacity constraints are not violated in expectation. Since we have assumed that the module degradation processes are independent, both within and across modules states and across time, the left side of each constraint represents the sum of independent random variables. Following from the law of large numbers, as the number of modules entered into repair increases, the resulting sum will more closely match the mean. This is another substantial benefit from the pooling affect of a centralized repair depot in which maintenance requirements from multiple locations are combined thus reducing the overall variability.

Execution While meeting the maintenance capacity requirements in expectation is a reasonable planning strategy, it becomes less viable as we transition from planning to maintenance operations. During the module maintenance scheduling and operations phase, we will continue to use the constraints (5.25) when generating a maintenance schedule. However, in addition to the expected maintenance usage, we will now consider the volatility inherent in the schedule. For example, consider two time periods t and t' in which the expected maintenance usage is equal. In the periods leading up to t, most of the scheduled maintenance actions are on high state modules. Due to their presumed high state, it is reasonable to assume that the variability around that state is relatively low. On the contrary, in the periods leading up to

t', a large number of mid to low state modules are scheduled for repair. Due to their lower state, there may be higher variability around the presumed state. In such a situation, maintenance schedules may plan for extra manpower or other variable capacity resources to augment the standing workforce in period t'.

This highlights a key aspect of the module submodels, the importance of surge repair capacity. This surge capacity can be used to address the stochastic events that occur and must be dealt with. Surge repair capacity is not modeled in the high level planning model due to the desire to use the planning model to set steady state manpower levels. By properly balancing the mix of maintenance actions over a long planning horizon, short term spikes or dips can be alleviated to some degree.

To formalize the notion of surge maintenance capacity, we consider the probability that the scheduled maintenance actions in a given time period exceed the scheduled maintenance capacity plus an inherent surge capability. In particular:

$$\mathbb{P}\left(\sum_{m\in\mathcal{M}}\sum_{\substack{m'\in\mathcal{M}:\\ m'\geq m}}\sum_{1\leq t'\leq t} a^{imm'}_{lt't}\mathbf{x^{imm'}_{lt'}}\right) > M^i_{lt}+\omega^i_{lt},$$

in which ω^i_{lt} represents the inherent surge capacity that is available in the module repair shop. By examining these probabilities across a number of module types, locations, and near-term time periods, a maintenance scheduler can reassign maintenance actions or request an augmented maintenance capacity. This is another benefit of maintaining modules in a centralized facility. By cultivating a flexible maintenance capacity comprised of adaptive equipment and highly skilled personnel, the depot can accommodate fluctuations in maintenance demand across module types and time periods without requiring a large permanent maintenance capacity for each separate module. The increased repair efficiencies prevalent in a centralized maintenance facility come at a price since, to function as planned, it depends heavily upon a reliable and available transportation network.

5.4 Implementing Nestedness

In Chapters 2 and 3, we introduced and demonstrated the importance of nestedness. In this section we will show how the idea of nestedness can be used in an augmented version of the planning model to help alleviate complex system assembly and module repair decisions. In addition, this demonstrates the importance of considering maintenance as a part of system design and development.

To define a (partially) nested schedule, we first order the modules based on the frequency of required maintenance, that is, according to something similar the f_i values defined for components in Chapter 2. We must establish the first module, which we call the minimum module, that has the most frequent maintenance requirement. While ranking the modules based on their maximum possible state is an obvious first choice, we must also consider the maintenance capacities and the time required to complete the repair activity. The module with the smallest maximum state may not be the best choice for the minimum module. There may be another module with a higher maximum state but the repair time needed to reach that maximum state is significant. To determine the minimum module we will augment the planning model developed in Section 5.1. To build a nested schedule, all modules will have a repair state, m_i, that is equal to an integer multiple of the minimum module's repair state, which we denote as m_1. That is $m_i = \kappa_i \cdot m_1$ for some positive integer κ_i. A module will only be repaired when it reaches the zero state. Thus, each time a system reaches the zero state, the module of type 1 in that system will also need repair along with some subset of the remaining modules.

As we indicate, the full linear program in Section 5.1 can be employed to determine module 1 and its corresponding maintenance level. This is accomplished by adding the following constraints to the formulation:

$$\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm'}} \leq \min_{t' \in [t, t+\tau_{lt}^{imm'})} M_{lt'}^{i} \cdot \sum_{\substack{\phi \in \mathcal{M}: \\ m' \bmod \phi = 0}} \mathbf{z}^{\phi}, \qquad \forall\, i \in \mathcal{I},\ m, m' \in \mathcal{M}, \\ l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.26}$$

$$\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm'}} \leq \min_{t' \in [t, t+\tau_{lt}^{imm'})} M_{lt'}^{i} \cdot (1 - \mathbf{z}^{\phi}), \qquad \forall\, i \in \mathcal{I},\ m, m', \phi \in \mathcal{M} : \phi < m \neq m',$$

$$l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.27}$$

$$\sum_{\phi \in \mathcal{M}} \mathbf{z}^{\phi} = 1, \tag{5.28}$$

$$\mathbf{z}^{\phi} \in \{0, 1\}. \tag{5.29}$$

In these constraints, $\mathbf{z}^{\phi}$ represents the choice for m_1, the minimum repair level that defines the nested schedule. Constraints (5.26) ensure that modules are only repaired to end states that are multiples of the chosen state. These constraints may need to be modified based on the exact relationship between modules states (i.e., if state 4 allows for twice as much usage as state 3, then states 3 and 4 are nested). Constraints (5.27) ensure that modules are not repaired until they cross the determined threshold. These modules can, however, be moved directly from the reparable to serviceable inventories.

While the formulation is now a mixed integer program, the relative size of $\mathcal{M}$ is quite small. In fact, we could further reduce the complexity of the problem by only considering states $m' \in \mathcal{M}$ that are reasonable end states (i.e., do not consider extremely low end state or high end states that have excessive repair times). In addition to determining module 1 and its associated maintenance level, we can use the solution to the above problem to determine the maintenance levels for all other modules by selecting the largest value of m' for which $\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm'}}$ is positive. An alternate construct would be to let the optimization problem choose the maintenance level for each of the modules. In this case, Constraints (5.26) will be replaced by three sets of constraints:

$$\mathbf{x}_{\mathbf{lt}}^{\mathbf{imm'}} \leq \min_{t' \in [t, t+\tau_{lt}^{imm'})} M_{lt'}^{i} \cdot \mathbf{z}^{\mathbf{im'}}, \qquad \forall\, i \in \mathcal{I},\ m, m' \in \mathcal{M},\ l \in \mathcal{L},\ 1 \leq t \leq T, \tag{5.30}$$

$$\mathbf{z}^{\mathbf{im'}} \leq \sum_{\substack{\phi \in \mathcal{M}: \\ m' \bmod \phi = 0}} \mathbf{z}^{\phi}, \qquad \forall\, i \in \mathcal{I},\ m' \in \mathcal{M}, \tag{5.31}$$

$$\sum_{m' \in \mathcal{M}} \mathbf{z}^{\mathbf{im'}} = 1, \qquad \forall\, i \in \mathcal{I} \tag{5.32}$$

$$\mathbf{z}^{\mathbf{im'}} \in \{0, 1\}. \tag{5.33}$$

The variable $\mathbf{z}^{\mathbf{im}}$ selects the repair level for module type i to be state m. Even with these additions, the number of binary variables remains quite small, no more than $|\mathcal{M}| \cdot (|I| + 1)$. Based on the solution times for the planning problem LP, this expanded version is still tractable. In addition, this expanded formulation will be solve very infrequently.

Since each serviceable module is in the same state or higher, namely state m_1, the network flow problem developed in Section 5.2.1 can be used to efficiently solve the system assembly portion of the problem. In addition, the predetermined maintenance levels simplify the module repair problem. For an individual location, the module repair problem can be solved by placing modules in maintenance as soon as capacity becomes available. This algorithm can be used even if the maintenance capacities are time varying. The same greedy algorithm can be used if the transportation times are assumed to be zero. Such an assumption is reasonable due to the relatively short transit times relative to the long module repair times.

By implementing the idea of nestedness, we are also able to better address the stochastic degradation of modules. In particular, we consider a problem in which modules are always repaired to a module specific state; but, when they arrive in the reparable inventory, they may be at a number of states. If a module of type i on a system that comes in for repair is above the m_1 threshold, then it is immediately transferred to the serviceable inventory. If it is below that threshold, it will be placed in the reparable inventory and repaired to state m' such that $\mathbf{z}^{\mathbf{im'}} = 1$. Such a policy would, however, require that all modules be inspected each time a system comes in for repair to ensure they are at or above the m_1 threshold. A more practical approach would be to leave modules with high expected states on systems that come in for repair. It should be removed for inspection only when it is close to reaching the m_1

threshold. As an example, a module that is initially installed in state $\kappa_i \cdot m_1$ could be left on the system for $\kappa_i - 2$ system repairs and then inspected thereafter. Such a procedure would ensure that systems have sufficient conservatism against stochastic degradation while not overwhelming the inspection and repair capacity.

5.5 Summary

Combined with the planning model development in the proceeding chapter, we have developed a framework for policy makers, maintenance planners, and front line schedulers to use in setting priorities and allocating resources for modular maintenance. In addition to the tradeoffs mentioned previously, this model can also be used to examine the effects of centralized/decentralized maintenance policies as well as the impacts of time varying demand. Current discussions within the Air Force revolve around the most advantageous way to schedule flights, and in particular which aircraft should be flown on a given day [13]. While past schedules have had little time variation, newer proposals vary the flight schedules significantly, front loading the beginning of a week and reducing flights later in the week based on expected aircraft maintenance requirements. Maintenance capacity, on the other hand, has remained relatively time invariant due to the long lead times required to train maintenance personnel. Using this modeling framework, policy makers and maintenance schedulers could use models to better understand the consequences changing these practices and other policy decisions that impact the interaction between inventory, maintenance, transportation, and operations.

Chapter 6

Conclusions and Future Work

Throughout this thesis we have focused on new models and algorithms for modular maintenance scheduling, particularly in critical systems that require preventative maintenance. Focusing first on the design considerations for modular maintenance, in Chapter 2 we formalized a model that can be used to tradeoff near term design and development costs with long term maintenance costs. To produce high quality, cyclic schedules, we developed two approximation algorithms, cycle rounding and shifted power-of-two, that have not only constant factor worst case guarantees, they both perform empirically close to optimal. In Chapter 3 we used two integer programming formulations to show that the worst case guarantee for the cycle rounding algorithm also holds for finite horizon instances. Also in Chapter 3, we utilized a linear program to extend the shifted power-of-two algorithm to the case in which the cycle limits are initially unknown. These efficient algorithms are first of their kind to address preventative modular maintenance scheduling while also addressing many of the underlying qualities needed for operational implementation.

An important extension of the Modular Maintenance Scheduling problem, as defined in Chapter 2, would be the inclusion of stochastically failing modules. By considering such modules, design teams could make tradeoffs between reliability and affordability. From an operational standpoint, this analysis could also yield insights for front-line maintainers for when they should perform preventative maintenance on cycle limited components after the failure of a stochastic component. Another

area for further study are the submodular cost functions used in Chapters 2 and 3. While the additive cost function on a directed acyclic graph as well as the downtime function are natural to consider in an aircraft engine setting, other submodular cost functions might be more applicable in other settings. In addition to identifying the functional forms of these cost functions, determining a systematic way to measure costs parameters is also needed. While the downtime function has a directly quantifiable meaning (i.e., the amount of time a systems spends out of service), the K_j costs used in the additive cost function can include variable costs for manpower and inventory in addition to fixed costs for facilities and equipment. In many applications these costs are quantified via dollar amounts which might not be practical or meaningful depending upon the situation. As mentioned in Chapter 2, one potential method to obtain maintenance costs is through the dual multipliers from a linear programming formulation that seeks to maximize the aggregate availability across a number of systems.

Shifting the focus to operations, in Chapter 4 we developed an in-depth mathematical model of module maintenance, system assembly, and transhipment in support of operations at numerous locations. This model captures the tradeoffs between maintenance capacity, shipment resources, and on-hand inventory. Due to the complexity of the model, in Chapter 5 we developed a decomposition strategy, and corresponding algorithms, to efficiently solve the problem. As discussed, the models and algorithms we develop in Chapters 4 and 5 are not restricted to a two-echelon environment. However, further work is needed to expand the model for use with multi-indenture systems. An initial method to accomplish this would be the inclusion of build and balance constraints for each level of indenture. However, such a strategy will greatly increase the size of the formulation and affect its tractability.

Since the planing model is solved as a linear programming problem, a robust linear programming formulation could be developed. By including robustness in to the formulation it could be used to account for stochastic system demand, stochastic module failure, or other random events. Robust optimization, especially the ideas used by Bertsimas and Sim [9], are especially attractive when with stochastic optimization

as robust optimization requires uncertainty sets versus a full characterization of the stochastic process.

A final area for exploration is the case when multiple modules share maintenance resources. In this context, the decomposition approaches would need to consider the interaction between modules when making maintenance decisions about each individual module type. Finally, many of the computational experiments were based on personal experience and data from recent reports. Implementation of this approach in an actual maintenance framework with real data might yield further insight into special structure, or other unique aspects of the application, that can be exploited to more efficiently solve the problem.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Notation

A.1 Modular Maintenance Scheduling Problem

$\mathcal{C}$	Set of components
$\mathcal{M}$	Set of all modules and components
T	Time Horizon
f_i	Cycle limit for component i
$K(\cdot)$	Submodular cost function
P_i	Dependency path for component i
K_j	The cost to remove and replace module $j \in \mathcal{M} \setminus \mathcal{C}$ or to repair component $j \in \mathcal{C}$
S_i	For a set of components $S \subset \mathcal{C}$, $S_i = S \cup 1, 2, \ldots, i-1$
K^i	Residual cost for component i, $K(\{1, 2, \ldots, i\}) - K(\{1, 2, \ldots, i-1\})$
$m(i)$	Parent of component i, component such that $K(\{1, 2, \ldots, i\}) - K(\{1, 2, \ldots, m(i)\}) = K^i$
$\widehat{f_i}$	Rounded cycle limits for cyclic maintenance schedule
R_i	Residual path of component i, $R_i = P_i \setminus \bigcup_{i' < i} P_{i'}$, equivalently $R_i = P_i \setminus P_{m(i)}$,
f_i^{CR}	Maintenance frequency of the cycle rounding algorithm for component i
δ	Shifted power-of-two rounding parameter
κ_i	Positive integer such that $2^{\kappa_i} \leq f_i < 2^{\kappa_i + 1}$

f_i^δ	Maintenance frequency for component i with rounding parameter δ
β_i	Real number in the interval $[1,2)$ such that $f_i = \beta_i \cdot 2^{\kappa_i}$
$\widetilde{f}_i$	Remaining cycles for component with minimum remaining cycles

A.2 The Graph Visiting Problem

G	Rooted and directed acyclic graph
$\mathcal{N}$	Node set of graph G
$\mathcal{A}_i$	Ancestors of node i, nodes on all paths from i to root including i
$K(S)$	Sum of cost for all nodes in set $S \subseteq \mathcal{N}$
S_t	Set of nodes visited at time period t
S_t^*	Set of nodes visited at period t in the optimal visitation schedule
f_i^r	Rounded down frequency requirement for node i
x_{st}^i	Variable indicating the visitation of node i at period s in preparation for period t
y_s^i	Variable indicating the visitation of node i at period s
x_ϕ^i	Variable assigning node i to a visitation frequency of $\phi \leq f_i$
D	Set of rounding a-priori rounding parameters
R_d	Fraction of total residual costs to assign to interval d

A.3 Modular Maintenance and System Assembly Model

Data

T	Time horizon
$\mathcal{I}$	Set of modules types with module 0 defined as the entire system
$\mathcal{L}$	Set of locations with location 0 being the depot
$\mathcal{M}$	Set of possible module states

$\mathcal{N}$	Set of possible system states
J	Set of deviation ranges for serviceable systems
S_{lt}^0	Number of serviceable systems available at location l in period t (Data for $t = 0$, variable for $t > 0$)
α_{ltj}	Cost function coefficients for serviceable systems shortfalls from demand at location l in time period t over the deviation range j
$\widetilde{\alpha}_{ltj}$	Cost function coefficients for serviceable systems shortfalls from reserve system level at location l in time period t over the deviation range j
$\mathcal{U}_{ltj}$	Upper bound on serviceable systems shortfalls from demand at location l in time period t over the deviation range j
$\widetilde{\mathcal{U}}_{ltj}$	Upper bound on serviceable systems shortfalls from reserve system level at location l in time period t over the deviation range j
R_{lt}^0	Number of reparable systems available at location l in period t (Data for $t = 0$, variable for $t > 0$)
B_{lt}^0	Number of serviceable systems that will return to service at location l in period t
A_{lt}^0	Number of serviceable systems that will leave service at location l in period t
D_{lt}	Demand for serviceable systems at location l in period t
U_{lt}	Maximum number of systems that can enter assembly at location l in period t
Q_{lt}	Minimum number of systems that can enter assembly at location l in period t
Q_{lt}^n	Minimum number of systems that can enter assembly at location l in period t with an assembled state of n

γ_{lt}	Number of periods required to assemble a system at location l beginning in time period t
ψ_{lt}^{n}	Approximate number of periods to exhaust the usable life of a system that was assemble in state n and used at location l beginning in period t
ξ_{lt}	Number of reserve serviceable systems desired at location l in period t, used to mitigate for random system failures
S_{lt}^{im}	Number of serviceable modules of type i in state m available at location l in period t (Data for $t = 0$, variable for $t > 0$)
R_{lt}^{im}	Number of reparable modules of type i in state m available at location l in period t (Data for $t = 0$, variable for $t > 0$)
B_{lt}^{im}	Number of serviceable modules of type i in state m that will return to service at location l in period t
A_{lt}^{im}	Number of reparable modules of type i in state m that will enter the reparable inventory at location l in period t
M_{lt}^{i}	Module repair capacity for modules of type i at location l in period t
$\tau_{lt}^{imm'}$	Number of periods required to repair a module of type i currently in state m to state m' at location l if repair begins in period t
$\nu_{ll'}$	Transportation time from location l to location l'
$\mu_{ll't}$	Total transportation capacity from location l to location l' for shipments that depart l in time period t
$\mu_{ll't}^{is}$	Transportation capacity from location l to location l' for shipments of serviceable modules of type i ($i = 0$ denotes serviceable systems) that depart l in time period t
$\mu_{ll't}^{ir}$	Transportation capacity from location l to location l' for shipments of reparable modules of type i ($i = 0$ denotes reparable systems) that depart l in time period t

s_{lt}^0 Number of serviceable systems in transport at the beginning of the planning horizon that will arrive at location l in time period t

r_{lt}^0 Number of reparable systems in transport at the beginning of the planning horizon that will arrive at location l in time period t

s_{lt}^{im} Number of serviceable modules of type i in state m in transport at the beginning of the planning horizon that will arrive at location l in time period t

r_{lt}^{im} Number of reparable modules of type i in state m in transport at the beginning of the planning horizon that will arrive at location l in time period t

Variables

$\mathbf{E_{ltj}}$ Tracks the difference between the serviceable systems inventory and the demand at location l in time period t for a specific range of deficiency j

$\widetilde{\mathbf{E}}_{\mathbf{ltj}}$ Tracks the difference between the serviceable systems inventory, the demand, and ξ_{lt} at location l in time period t for a specific range of deficiency j

$\mathbf{x_{lt}^{imm'}}$ Number of reparable modules of type i currently in state m to repair to state m' at location l beginning in period t

$\mathbf{y_{lt}^{0n}}$ Number of systems to assemble in state n at location l beginning in time period t

$\mathbf{y_{lt}^{im}}$ Number of serviceable modules of type i in state m to use in assembling systems at location l beginning in time period t

$\mathbf{w_{lt}^{imn}}$ Number of serviceable modules of type i in state m to assemble into systems yielding a system state of n at location l beginning in time period t

$\mathbf{r^0_{ll't}}$	Number of reparable systems to ship from location l to location l' in period t
$\mathbf{s^{0n}_{ll't}}$	Number of serviceable systems in state n to ship from location l to location l' in period t
$\mathbf{r^{im}_{ll't}}$	Number of reparable modules of type i in state m to ship from location l to location l' in period t
$\mathbf{s^{im}_{ll't}}$	Number of serviceable modules of type i in state m to ship from location l to location l' in period t

Hierarchical Model

$\overline{S^{0n}_{lt}}$	Serviceable system target for system state n at location l in time period t
$\overline{S^{im}_{lt}}$	Serviceable module target for modules of type i in state m at location l in time period t
$\mathfrak{G}^{0n}_{lt}$	Cost function for target deviations from $\overline{S^{0n}_{lt}}$
$\mathfrak{G}^{im}_{lt}$	Cost function for target deviations from $\overline{S^{im}_{lt}}$
$\underline{y^{im}_{lt}}$	Modules of type i in state m needed for system assembly at location l at time period t

Appendix B

Full Modular Maintenance and System Assembly Formulation

$$\text{minimize} \sum_{l \in \mathcal{L}} \sum_{j in J} \sum_{1 \leq t \leq T} \left(\alpha_{ltj} \cdot \mathbf{E}_{ltj} + \widetilde{\alpha}_{ltj} \cdot \widetilde{\mathbf{E}}_{\mathbf{ltj}} \right) \tag{P}$$

subject to

Objective function constraints

Deviation from demand

$$\mathbf{S}^{\mathbf{0}}_{\mathbf{lt}} + \sum_{j \in J} \mathbf{E}_{\mathbf{ltj}} \geq D_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.1}$$

Deviation from desired serviceable system reserve

$$\mathbf{S}^{\mathbf{0}}_{\mathbf{lt}} + \sum_{j \in J} \left(\mathbf{E}_{\mathbf{ltj}} + \widetilde{\mathbf{E}}_{\mathbf{ltj}} \right) \geq \xi_{lt} + D_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.2}$$

Deviation range upper bounds

$$\mathbf{E}_{\mathbf{ltj}}, \leq \mathcal{U}_{ltj} \qquad \forall\, l \in \mathcal{L},\ j \in J,\ 1 \leq t \leq T, \tag{B.3}$$

$$\widetilde{\mathbf{E}}_{\mathbf{ltj}}, \leq \widetilde{\mathcal{U}}_{ltj} \qquad \forall\, l \in \mathcal{L},\ j \in \widetilde{J}, \qquad 1 \leq t \leq T, \tag{B.4}$$

Problem constraints

Reparable systems inventory: inventory from previous period, plus systems coming out of service and in transport, minus systems assembled, plus net inflow of reparable systems

$$\begin{aligned}
\mathbf{R}^{\mathbf{0}}_{\mathbf{lt}} = \mathbf{R}^{\mathbf{0}}_{\mathbf{l(t-1)}} + A^0_{lt} + r^0_{lt} - \sum_{n\in\mathcal{N}} \mathbf{y}^{\mathbf{0n}}_{\mathbf{lt}} + \\
\sum_{n\in\mathcal{N}} \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \left(\mathbf{y}^{\mathbf{0n}}_{\mathbf{lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s}^{\mathbf{0n}}_{\mathbf{ll'(t'+\gamma_{lt'})}} \right) + \\
\sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\leq t'\leq t:\\ t'+\nu_{l'l}+\psi^n_{l(t'+\nu_{l'l})}=t}} \mathbf{s}^{\mathbf{0n}}_{\mathbf{l'lt'}} + \\
\sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\leq t'\leq t:\\ t'+\nu_{l'l}=t}} \mathbf{r}^{\mathbf{0}}_{\mathbf{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r}^{\mathbf{0}}_{\mathbf{ll't}} \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T,
\end{aligned} \tag{B.5}$$

Serviceable systems inventory: initial inventory, inflow from assembly/test and other locations, assemblies during the planning period minus systems shipped to other locations, plus inflow of systems from other locations

$$\begin{aligned}
\mathbf{S}^{\mathbf{0}}_{\mathbf{lt}} = \mathbf{S}^{\mathbf{0}}_{\mathbf{l(t-1)}} - A^0_{lt} + B^0_{lt} + s^0_{lt} + \\
\sum_{n\in\mathcal{N}} \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}=t}} \mathbf{y}^{\mathbf{0n}}_{\mathbf{lt'}} - \sum_{n\in\mathcal{N}} \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \mathbf{y}^{\mathbf{0n}}_{\mathbf{lt'}} + \qquad \forall\, l \in \mathcal{L}:\ l \neq 0, \\
\sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\leq t'\leq t:\\ t'+\nu_{l'l}=t}} \mathbf{s}^{\mathbf{0n}}_{\mathbf{l'lt'}} - \sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s}^{\mathbf{0n}}_{\mathbf{ll't}}, \qquad 1 \leq t \leq T,
\end{aligned} \tag{B.6}$$

Test stand capacity

$$\sum_{n\in\mathcal{N}} \mathbf{y_{lt}^{0n}} \leq U_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.7}$$

Maintenance proficiency requirement

$$\sum_{n\in\mathcal{N}} \mathbf{y_{lt}^{0n}} \geq Q_{lt}, \qquad \forall\, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.8}$$

State specific maintenance proficiency requirement

$$\mathbf{y_{lt}^{0n}} \geq Q_{lt}^{n}, \qquad \forall\, n \in \mathcal{N}, l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.9}$$

Reparable module inventory: inventory from previous period, plus modules coming out of service and inflow from other locations, plus modules from planning period coming out of service (minus those shipped to other locations), minus modules placed in repair, plus net inflow of reparable modules

$$\mathbf{R_{lt}^{im}} = \mathbf{R_{l(t-1)}^{im}} + A_{lt}^{im} + r_{lt}^{im} + \sum_{n\in\mathcal{N}} \left(\sum_{\substack{m'\in\mathcal{M}:\\ m'-n=m}} \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}+\psi^{n}_{l(t'+\gamma_{lt'})}=t}} \mathbf{w_{lt'}^{im'n}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s_{ll'(t'+\gamma_{lt'})}^{0n}} \right) - \sum_{\substack{m'\in\mathcal{M}:\\ m'\geq m}} \mathbf{x_{lt}^{imm'}} + \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\leq t'\leq t:\\ t'+\nu_{l'l}=t}} \mathbf{r_{l'lt'}^{im}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r_{ll't}^{im}}, \qquad \forall\, i \in \mathcal{I},\ m \in \mathcal{M}:\ m \neq 0,\ l \in \mathcal{L},\ 1 \leq t \leq T, \tag{B.10}$$

Reparable module inventory: inventory from previous period, plus modules coming out of service and inflow from other locations, plus modules from planning period coming out of service (minus those shipped to other locations), minus modules placed in repair, plus net inflow of reparable modules, plus modules coming out of service from serviceable systems shipped in from other locations

$$\begin{aligned}
\mathbf{R^{i0}_{lt}} = \mathbf{R^{i0}_{l(t-1)}} + A^{i0}_{lt} + r^{i0}_{lt} + \\
\sum_{n\in\mathcal{N}} \left(\sum_{\substack{m'\in\mathcal{M}:\\ m'-n=0}} \sum_{\substack{1\le t'\le t:\\ t'+\gamma_{lt'}+\psi^n_{l(t'+\gamma_{lt'})}=t}} \mathbf{w^{im'n}_{lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{0n}_{ll'(t'+\gamma_{lt'})}} \right) - \\
\sum_{m'\in\mathcal{M}} \mathbf{x^{i0m'}_{lt}} + \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{r^{i0}_{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{r^{i0}_{ll't}} + \\
\sum_{n\in\mathcal{N}} \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}+\psi^n_{l(t'+\nu_{l'l})}=t}} \mathbf{s^{0n}_{l'lt'}}, \qquad 1\le t\le T,
\end{aligned} \qquad \forall\, i\in\mathcal{I},\ l\in\mathcal{L}, \tag{B.11}$$

Serviceable module inventory: inventory from previous period, plus modules coming out of repair and inflow from other locations, minus modules used in system assembly, plus modules coming out of repair, plus net inflow of reparable modules

$$\begin{aligned}
\mathbf{S^{im}_{lt}} = \mathbf{S^{im}_{l(t-1)}} + B^{im}_{lt} + s^{im}_{lt} - \mathbf{y^{im}_{lt}} + \\
\sum_{\substack{m'\in\mathcal{M}:\\ m'\le m}} \sum_{\substack{1\le t'\le t:\\ t'+\tau^{im'm}_{lt'}=t}} \mathbf{x^{im'm}_{lt'}} + \\
\sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \sum_{\substack{1\le t'\le t:\\ t'+\nu_{l'l}=t}} \mathbf{s^{im}_{l'lt'}} - \sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s^{im}_{ll't}}, \qquad 1\le t\le T,
\end{aligned} \qquad \forall\, i\in\mathcal{I},\ m\in\mathcal{M},\ l\in\mathcal{L}, \tag{B.12}$$

Module repair capacity

$$\sum_{m\in\mathcal{M}} \sum_{\substack{m'\in\mathcal{M}:\\ m'\ge m}} \sum_{\substack{1\le t'\le t:\\ t'+\tau^{imm'}_{lt'}>t}} \mathbf{x^{imm'}_{lt'}} \le M^i_{lt} - \sum_{m\in\mathcal{M}} \sum_{t<t'\le T} B^{im}_{lt'}, \qquad \forall\, l\in\mathcal{L},\ i\in\mathcal{I},\ 1\le t\le T, \tag{B.13}$$

System build constraints

$$\sum_{\substack{m\in\mathcal{M}:\\ m\geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{0n}}, \qquad \forall\, l\in\mathcal{L},\ i\in\mathcal{I},\ n\in\mathcal{N},\ 1\leq t\leq T, \tag{B.14}$$

Module balance constraints

$$\sum_{\substack{n\in\mathcal{N}:\\ m\geq n}} \mathbf{w}_{\mathbf{lt}}^{\mathbf{imn}} = \mathbf{y}_{\mathbf{lt}}^{\mathbf{im}}, \qquad \forall\, l\in\mathcal{L},\ i\in\mathcal{I},\ m\in\mathcal{M},\ 1\leq t\leq T, \tag{B.15}$$

Serviceable systems only shipped coming out of assembly and test

$$\sum_{\substack{l'\in\mathcal{L}:\\ l'\neq l}} \mathbf{s}_{\mathbf{ll't}}^{\mathbf{0n}} \leq \sum_{\substack{1\leq t'\leq t:\\ t'+\gamma_{lt'}=t}} \mathbf{y}_{\mathbf{lt'}}^{\mathbf{0n}}, \qquad \forall\, n\in\mathcal{N},\ l\in\mathcal{L},\ 1\leq t\leq T, \tag{B.16}$$

Reparable systems flow

$$\mathbf{r}_{\mathbf{ll't}}^{\mathbf{0}} \leq \mu_{ll't}^{0r}, \qquad \forall\, l,l'\in\mathcal{L}:\ l\neq l',\ 1\leq t\leq T, \tag{B.17}$$

Serviceable systems flow

$$\sum_{n\in\mathcal{N}} \mathbf{s}_{\mathbf{ll't}}^{\mathbf{0n}} \leq \mu_{ll't}^{0s}, \qquad \forall\, l,l'\in\mathcal{L}:\ l\neq l',\ 1\leq t\leq T, \tag{B.18}$$

Reparable module flow

$$\sum_{m\in\mathcal{M}} \mathbf{r}_{\mathbf{ll't}}^{\mathbf{im}} \leq \mu_{ll't}^{ir}, \qquad \forall\, l,l'\in\mathcal{L}:\ l\neq l',\ i\in\mathcal{I},\ 1\leq t\leq T, \tag{B.19}$$

Serviceable module flow

$$\sum_{m \in \mathcal{M}} \mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}} \leq \mu^{is}_{ll't}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l',\ i \in \mathcal{I},\ 1 \leq t \leq T, \tag{B.20}$$

Combined flow

$$\sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} \left(\mathbf{s}^{\mathbf{im}}_{\mathbf{ll't}} + \mathbf{r}^{\mathbf{im}}_{\mathbf{ll't}}\right) + \sum_{n \in \mathcal{N}} \mathbf{s}^{\mathbf{0n}}_{\mathbf{ll't}} + \mathbf{r}^{\mathbf{0}}_{\mathbf{ll't}} \leq \mu_{ll't}, \qquad \forall\, l, l' \in \mathcal{L} :\ l \neq l',\ 1 \leq t \leq T, \tag{B.21}$$

Nonnegativity and integrality of the decision variables

$$\mathbf{E}, \widetilde{\mathbf{E}}, \mathbf{R}, \mathbf{S}, \mathbf{y}, \mathbf{x}, \mathbf{w}, \mathbf{r}, \mathbf{s} \in \mathbb{Z}^{+}. \tag{B.22}$$

Appendix C

Set Cover Reduction to Time Varying Cost Problem

Consider an instance of unweighted set cover with a ground set of elements $\mathcal{C} = \{e_1, \ldots, e_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\} \subseteq 2^{\mathcal{C}}$. We create an instance of the modular maintenance scheduling problem as follows:

- The planning horizon is of length $m + 1$, where the time periods are numbered $0, \ldots, m$.
- There is one component for every element in $\mathcal{C}$, each with a cycle limit of m.
- For every time period, the cost of the single shared module is 1. Also, for all components $i \in \mathcal{C}$:

    1. We make sure that none of the components are maintained at period 0, by setting $K_i^0 = \infty$.

    2. We create a one-to-one correspondence between maintaining components corresponding to the elements in S_t at time period t and between picking the subset S_t in the set cover instance. This is achieved by setting $K_i^t = 0$ for every t such that $e_i \in S_t$, and $K_i^t = \infty$ otherwise.

Note that since components cannot be maintained at period 0 within incurring infinite cost, and since the cycle limit of every component is m, each component must

be maintained exactly once over the planning horizon between period 1 and m, as one unit of its cycle limit is inevitably exhausted in period 0. With this observation is place, we can see that the optimal maintenance schedule will have a cost equal to the minimum set cover. This implies that the modular maintenance scheduling problem with time varying costs cannot be approximated within a constant factor as set cover cannot be approximated better than $O(\ln(n))$, where n is the number of ground elements, unless P=NP [8, 39].

Bibliography

[1] 2010. Defense logistics agency fact sheet. Fact Sheet.

[2] 2011. Airman magazine: The book 2011. Tech. rep., United States Air Force.

[3] AFI. 1998. Maintenance: Two level maintenance and regional repair of air force weapon systems and equipment. Air Force Instruction 21-129, United States Air Force.

[4] AFI. 2004. Aircraft standard utilization rate procedures. Air Force Instruction 11-103, United States Air Force.

[5] AFPD. 2004. Flying hour program. Air Force Policy Directive 11-1, United States Air Force.

[6] Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

[7] Amouzegar, M.A., L.A. Galway, A. Geller. 2002. Alternatives for jet engine intermediate maintenance. Tech. Rep. RB-82-AF, RAND, Santa Monica, CA.

[8] Arora, S., M. Sudan. 2003. Improved low-degree testing and its applications. *Combinatorica* **23** 365–426.

[9] Bertsimas, D., M. Sim. 2004. The price of robustness. *Oper. Res.* **52** 35–53.

[10] Bitran, G.R., E.A. Haas, A.C. Hax. 1982. Hierarchical production planning: A two-stage system. *Oper. Res.* **30**(2) 232–251.

[11] Bitran, G.R., D. Tirupati. 1993. Hierarchical production planning. *Handbooks in operations research and management science* **4** 523–568.

[12] Chenoweth, M. E., C. A. Gammich. 2006. The f100 engine purchasing and supply chain management demonstration: Findings from air force spend analyses. Tech. Rep. MG-424, RAND, Santa Monica, CA.

[13] Cho, P.Y. 2011. Optimal scheduling of fighter aircraft maintenance. Master's thesis, Massachusetts Institute of Technology.

[14] Crowston, W.B., A. Henshaw, M.H. Wagner. 1972. A comparison of exact and heuristic routines for lot-size determination in multi-stage assembly systems. *AIIE Trans.* **4** 313–317.

[15] Deshpande, V., A.V. Iyer, R. Cho. 2006. Efficient supply chain management at the US Coast Guard using part-age dependent supply replenishment policies. *Oper. Res.* **54**(6) 1028.

[16] Díaz, A. 1995. Multi-echelon inventory models for repairable items. Ph.D. thesis, University of Maryland.

[17] Díaz, A., M.C. Fu. 1997. Models for multi-echelon repairable item inventory systems with limited repair capacity. *Eur. J. Oper. Res.* **97**(3) 480 – 492.

[18] Everingham, K., G. Polaski, F. Riedlin, M. Shirk, V. Deshpande, A.V. Iyer. 2008. Operations Research Enhances Supply Chain Management at the US Coast Guard Aircraft Repair and Supply Center. *Interfaces* **38**(1) 61.

[19] Federgruen, A., M. Queyranne, Y.S. Zheng. 1992. Simple power-of-two policies are close to optimal in a general class of production/distribution networks with general joint setup costs. *Math. Oper. Res.* **17**(4) 951–963.

[20] Federgruen, A., Y.S. Zheng. 1993. Optimal power-of-two replenishment strategies in capacitated general production/distribution networks. *Management Sci.* **39**(6) 710–727.

[21] Forbes, J.A., P.P. Wyatt. 1975. Optimal replacement policy for the F-15 aircraft engine modules. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH.

[22] Geller, A., D. George, R.S. Tripp, M.A. Amouzegar, C.R. Roll. 2004. Supporting air and space expeditionary forces: Analysis of maintenance forward support location operations. Tech. Rep. MG-151-AF, RAND, Santa Monica, CA.

[23] Graves, S.C. 1982. Using lagrangean techniques to solve hierarchical production planning problems. *Management Sci.* **28**(3) 260–275.

[24] Graves, S.C. 1985. A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management Sci.* **31**(10) 1247–1256.

[25] Hillestad, R. J., M. J. Carrillo. 1980. Models and techniques for recoverable item stockage when demand and the repair process are nonstationary–part 1: Performance measurement. Tech. Rep. N-1482-AF, RAND, Santa Monica, CA.

[26] Keating, E.G., D. Snyder, M.C. Dixon, E.N. Lorego. 2005. Aging aircraft repair-replacement decisions with depot-level capacity as a policy choice variable. Tech. Rep. MG-241-AF, RAND, Santa Monica, CA.

[27] Kobbacy, K.A.H., D.N.P. Murthy, R.P. Nicolai, R. Dekker. 2008. Optimal maintenance of multi-component systems: A review. Hoang Pham, ed., *Complex System Maintenance Handbook*. Springer Series in Reliability Engineering, Springer London, 263–286.

[28] Levi, R., R.O. Roundy, D.B. Shmoys. 2006. Primal-Dual Algorithms for Deterministic Inventory Problems. *Math. Oper. Res.* **31**(2) 267–284.

[29] Loredo, E.N., R.A. Pyles, D. Snyder. 2007. Programmed depot maintenance capacity assessment tool: Workloads, capacity, and availability. Tech. Rep. MG-519-AF, RAND, Santa Monica, CA.

[30] Maxwell, M.L., J.A. Muckstadt. 1985. Establishing consistent and realistic reorder intervals in production-distribution systems. *Oper. Res.* **33**(6) 1316–1341.

[31] McGarvey, R.G., M. Carrillo, D.C. Cato, J.G. Drew, T. Lang, K.F. Lynch, A.L. Maletic, H.G. Massey, J.M. Masters, R.A. Pyles, R. Sanchez, J.M. Sollinger, B. Thomas, R.S. Tripp, B.D. Van Roo. 2009. Analysis of the Air Force Logistics Enterprise: Evaluation of Global Repair Network Options for Supporting the F-16 and KC-135. Tech. Rep. MG-872-AF, RAND, Santa Monica, CA.

[32] Miller, B. L., M. Modarres-Yazdi. 1978. The distribution of recoverable inventory items from a repair center when the number of consumption centers is large. *Naval Research Logistics Quarterly* **25**(4) 597–604.

[33] Miller, B.L. 1974. Dispatching from depot repair in a recoverable item inventory system: On the optimality of a heuristic rule. *Management Sci.* **21** 316–325.

[34] Muckstadt, J.A. 1973. A model for a multi-item, multi-echelon, multi-indenture inventory system. *Management Sci.* **20**(4-Part-I) 472–481.

[35] Muckstadt, J.A. 1976. The consolidated support model (csm): A three-echelon, multi-item model for recoverable items. Tech. Rep. R-1923-PR, RAND, Santa Monica, CA.

[36] Muckstadt, J.A. 1978. Some approximations in multi-item, multi-echelon inventory systems for recoverable items. *Naval Research Logistics Quarterly* **25**(3) 377–394.

[37] Muckstadt, J.A. 1980. Comparative adequacy of steady-state versus dynamic models for calculating stockage requirements. Tech. Rep. R-2636-AF, RAND, Santa Monica, CA.

[38] Muckstadt, J.A. 2005. *Analysis and algorithms for service parts supply chains.* Springer Series in Operations Research and Financial Engineering, Springer.

[39] Raz, R., S. Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. STOC '97, ACM, New York, NY, USA, 475–484.

[40] Roundy, R. 1985. 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems. *Management Sci.* **31**(11) 1416–1430.

[41] Roundy, R. 1989. Rounding off to powers of two in continuous relaxations of capacitated log sizing problems. *Management Sci.* **35**(12) 1433–1442.

[42] Sherbrooke, C.C. 1968. Metric: A multi-echelon technique for recoverable item control. *Oper. Res.* **16**(1) 122–141.

[43] Sherbrooke, C.C. 1986. Vari-metric: Improved approximations for multi-indenture, multi-echelon availability models. *Oper. Res.* **34** 311–319.

[44] Sherbrooke, C.C. 2004. *Optimal inventory modeling of systems: multi-echelon techniques*. International series in operations research & management science, Kluwer Academic.

[45] Teo, C.P., D. Bertsimas. 2001. Multistage lot sizing problems via randomized rounding. *Oper. Res.* **49**(4) 599–608.

[46] Tripp, R.S., R.G. McGarvey, B.D. Van Roo, J.M. Masters, J.M. Sollinger. 2010. A Repair Network Concept for Air Force Maintenance: Conclusions from Analysis of C-130, F-16, and KC-135 Fleets. Tech. Rep. MG-919-AF, RAND, Santa Monica, CA.

[47] van Dijkhuizen, G. 2000. Maintenance grouping in multi-step multi-component production systems. M. Ben-Daya, S. O. Duffuaa, A. Raouf, eds., *Maintenance, Modeling, and Optimization*. Klumer Academic, Norwell, MA, 283–306.

[48] van Dijkhuizen, G., A. van Harten. 1997. Optimal clustering of frequency-constrained maintenance jobs with shared set-ups. *Eur. J. Oper. Res.* **99**(3) 552 – 564.

[49] Wang, H., H. Pham. 2006. *Reliability and optimal maintenance*. Springer Verlag.